UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Aprendizaje y corrección de errores en sistemas de seguimiento basados en redes convolucionales siamesas

Máster Universitario en Ingeniería de Telecomunicación

Autor: Iglesias Arias, Álvaro

Tutor: Escudero Viñolo, Marcos

Departamento de Tecnología Electrónica y de las Comunicaciones

FECHA: Junio, 2021

APRENDIZAJE Y CORRECIÓN DE ERRORES EN SISTEMAS DE SEGUIMIENTO BASADOS EN REDES CONVOLUCIONALES SIAMESAS

Autor: Álvaro Iglesias Arias
Tutor: Marcos Escudero Viñolo
Ponente: Jesús Bescós Cano

Dpto. Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2021

Este trabajo ha sido parcialmente financiado por el gobierno de España a través del proyecto TEC2017-88169-R MobiNetVideo.

Resumen

Este Trabajo de Fin de Máster tiene como objetivo el estudio y mejora de los sistemas de seguimiento (*trackers*) basados en redes neuronales siamesas frente a diferentes problemas clásicos en el seguimiento de objetos (*tracking*) como pueden ser las oclusiones o los distractores (presencia de objetos idénticos al que se desea seguir en la misma secuencia).

Desde la explosión de los sistemas de aprendizaje profundo hace unos años y el aumento en complejidad y tamaño de las bases de datos que esto ha conllevado no hay prácticamente campo de la ingeniería que no se haya visto afectado por estos avances y, sin duda, el campo por excelencia que representa estos cambios ha sido el de la visión artificial o *computer vision.* Este es un campo grande con gran variedad de aplicaciones distintas, cada una con sus propios desafíos. De entre todas estas aplicaciones una de las más complejas es la del seguimiento de objetos debido a la variedad de situaciones posibles que requieren de sistemas capaces de adaptarse a cualquier situación.

Para ello, en este trabajo se han propuesto dos algoritmos basados en seguimiento hacia atrás (o *backtracking*) e implementados en *Matlab* con el objetivo de paliar los problemas de los *trackers* frente a dos problemas concretos; las oclusiones y los distractores. Estos sistemas funcionan como módulos que se pueden añadir sobre la salida de las redes siamesas empleadas para el seguimiento en diferentes *trackers* y refinan sus predicciones, lo cual hace de estos sistemas especialmente versátiles ya que puede ser empleados en diferentes situaciones y *trackers*.

También, para validar correctamente los resultados obtenidos ha sido necesario generar un dataset propio, a partir de un subconjunto de videos obtenidos de otros datasets, con todos estos problemas etiquetados, tanto a nivel de video como a nivel de *frame*.

Finalmente se han comprobado los diferentes resultados obtenidos en cada una de estas situaciones utilizando el dataset creado y se han analizado las diferentes mejoras, así como los problemas encontrados.

Palabras Clave

Aprendizaje Profundo, Visión Artificial, Seguimiento de Objetos, Tracker, Red Neuronal Siamesa, Backtracking, Matlab, Oclusiones, Distractores, Correlación, Video Object Tracking.

Abstract

This Master's Thesis aims to study and refine trackers based on Siamese neural networks in the face of classic challenges in the field of object tracking, such as occlusions or distractors (presence on the scene of object with the same appearance as the target object).

Since the explosion of deep learning techniques a few years ago and the increase in complexity and size of the available datasets there is no field of engineering that has not been affected by this. One of the most representative examples of this changes can be found in the field of the computer vision. There are so much different applications in this ambit each one with its one challenges. From all these applications one of the most challenging is object tracking due to the diversity of possible situations that can be encountered. Because of this tracker systems must be capable of adapt to multiple situations with ease

In this project two different algorithms have been proposed and implemented in *Matlab*, both based in the technique known as backtracking (or backward tracking). Their objective is to mitigate the problems that arise in multiple trackers in presence of two different events: target occlusions and the appearance of distractors. These systems work as modules that can be attached to the end of the siamese neural networks that conforms the core of the tracker and use their output to refine the results obtained. This system-module approach makes this work flexible and open possibilities of improvements in multiple trackers that conform the state of the art.

Also, to validate the results obtained a dataset with labeled occlusions at frame level has been created from a subset of videos obtained from other datasets. These videos and their events have also been tagged at video level.

Lastly, results have been checked for each one of these events using our own dataset. Improvements and failures of each category have been analysed and commented also the problems found.

Keywords

Deep Learning, Computer Vision, Object Tracking, Tracker, Siamese Neural Networks, Backtracking, Matlab, Oclusion, Distractor, Correlation, Correlation Filter, Video Object Tracking.

A mi abuela, gracias por todo.

TABLA DE CONTENIDO

1 INTRODUCCIÓN

1.1 MOTIVACIÓN

Los sistemas de *tracking* (o seguimiento) de objetos por video constituyen una de las áreas de la visión artificial que más ha evolucionado en los últimos años. Avanzando de la mano del desarrollo de los sistemas de aprendizaje profundo (*deep learning*), el estado del arte ha llegado a cotas que superan ampliamente cualquier resultado conseguido previamente a este salto tecnológico [1] [2] [3]. Sin embargo, siguen existiendo diversos desafíos en esta área que aun han de ser solventados. Entre ellos, cabe destacar los problemas de rendimiento, las dificultades al realizar el seguimiento en presencia de otros objetos similares o las dificultades para detectar cambios de apariencia. Es por ello por lo que en este trabajo nos hemos querido centrar en el análisis e intento de mitigación de unos de estos problemas, muy común en este tipo de tarea: la presencia de oclusiones y distractores (objetos de apariencia idéntica al original y que se encuentran presentes en la misma escena) que dificultan e incluso impiden el correcto seguimiento del objeto [4].

Para ello se propone desarrollar un sistema que, haciendo uso de las características del objeto a seguir extraídas por el *tracker*, nos permita refinar y corregir diferentes fallos debidos a estas oclusiones parciales y distractores. Además, este sistema se diseñará de manera que pueda unirse, a modo de módulo, a diferentes sistemas ya existentes de cara a mejorar sus resultados.

El sistema propuesto hace uso de una técnica conocida como *backtracking* la cual nos permite comprobar si la posición del objeto que se ha seleccionado en la imagen del video bajo análisis es correcta yendo marcha atrás, es decir, realizando el tracking de la posición en el *frame* actual hasta un *frame* anterior. Teóricamente si la posición ha sido correctamente seleccionada este seguimiento hacia atrás nos llevara a la posición original del objeto en el *frame* anterior. De esta forma siguiendo este método podemos comprobar, en los casos que sea necesario, si nuestra posición es correcta o no basándonos en si el sistema es capaz de volver al mismo objeto en el *frame* anterior.

Adicionalmente, para la realización de este trabajo y, ante la ausencia de *datasets* con la información requerida para la creación de sistemas como los comentados, se llevará a cabo la creación de un *dataset* propio y robusto con etiquetas de oclusión a nivel de imagen en cada video. También se realizará un segundo conjunto de etiquetas para este *dataset* con anotaciones tanto de oclusiones (parciales y completas) como de distractores a nivel de video. Esto nos permitirá validar y comprobar el comportamiento de los diferentes sistemas frente a estos eventos con mucha mayor precisión dando lugar así a resultados más completos.

1.2 OBJETIVOS

Los diferentes objetivos de este trabajo se pueden resumir en:

- Creación de una base de datos de imágenes ocluidas robusta que permita el entrenamiento y la validación de los diferentes sistemas implementados.

- Desarrollo de un sistema que permita, mediante el uso del *backtracking*, aumentar la robustez y fiabilidad de diferentes sistemas de *tracking* frente a oclusiones parciales y distractores.
- Integración y evaluación de este sistema sobre un *tracker* del estado del arte y cuantificación de las mejoras obtenidas ante la presencia de diferentes retos como pueden ser las oclusiones parciales o los distractores.

1.3 ORGANIZACIÓN DE LA MEMORIA

Este trabajo se encuentra organizado de la siguiente manera:

- **Introducción**: En este apartado se expone brevemente la temática del trabajo y sus diferentes objetivos. Este apartado busca guiar al lector a lo largo de esta memoria y darle una perspectiva general sobre los diferentes asuntos que se tratan en ella.
- **Estado del arte:** En este apartado se describe la literatura más relevante en el ámbito del seguimiento de objetos, no solo sistemas actuales si no también otros sistemas anteriores que permiten contextualizar el estado del arte actual. También se explicarán como otros trabajos se han enfrentado al problema de las oclusiones y las soluciones que han propuesto.
- **Diseño y desarrollo**: Aquí se explican los sistemas creados para este trabajo, su motivación, funcionamiento, características y parámetros.
- **Resultados**: En esta sección se exponen las diferentes evaluaciones experimentales realizadas, así como los resultados cuantitativos obtenidos para cada una de ellas. Tras mostrar los resultados se analizarán estos, se desglosarán en diferentes partes para dar una visión más completa y se estudia su comportamiento frente a las diferentes bases de datos empleadas.
- **Conclusiones y trabajo futuro:** En este apartado se resumen las conclusiones alcanzadas en este trabajo y se enumeran posibles líneas futuras de investigación que deja abiertas este proyecto.

2 ESTADO del ARTE

2.1 TRACKING

Podemos definir el *tracking* (o seguimiento) de objetos en video como el problema que busca localizar la posición de uno o varios objetos (conocidos como *target* u objetivos) en movimiento a lo largo del tiempo usando una cámara. Tiene una gran variedad de aplicaciones: interacción humano-máquina [5], usos en seguridad y vigilancia [6], realidad aumentada [7], control del tráfico [8] o conducción autónoma [9] entre otras.

Es un tipo de tarea muy intensiva debido a la gran cantidad de información que se ha de procesar en tiempo real para poder realizar el seguimiento on-line. Esto se puede mitigar haciendo uso de imágenes con una menor resolución o con una captando una menor cantidad de imágenes por segundo, pero esto tiene el inconveniente de dificultar la tarea debido a la perdida de información que se produce en consecuencia. Es por ello por lo que en este tipo de sistemas no solo se valora la precisión con la que se siguen los diferentes objetos si no también la velocidad a la que se puede realizar este seguimiento [10]: un sistema con un rendimiento pobre o una eficiencia baja no será capaz de cumplir con las condiciones necesarias para operar en tiempo real.

Otra de las características que definen un sistema de tracking es su capacidad de adaptarse a cualquier objeto. Deben ser capaces de seguir cualquier tipo sin necesidad de disponer de ninguna información a priori sobre este, utilizando una única muestra o imagen modelo del objeto, habitualmente su apariencia en el primer *frame* del video. Para ello, estos sistemas abstraen las diferentes propiedades del objeto *target* mediante una serie de características que obtienen a partir de la imagen disponible (el tipo de características utilizadas para representar al *target* serán diferentes para cada sistema, cada una con sus ventajas e inconvenientes). Pero este requerimiento de que el sistema ha de ser capaz de generalizar a cualquier tipo de objeto hace que la selección de características empleadas para representar al objeto *target* sea especialmente relevante, puesto que las características tienen que permitir diferenciarlo de cualquier otro objeto, tanto similar (incluso de la misma clase) como diferente, así como del propio fondo de la escena.

Otro de los problemas clásicos de estos sistemas es el cambio en las características del objeto a lo largo de la secuencia. Estas variaciones incluyen cambios de tamaño según el objeto se acerca o se aleja, cambios de color según cambia la iluminación o la posición del objeto e incluso cambios de forma debido al movimiento de la cámara respecto del objeto o al propio movimiento de éste a lo largo de la secuencia. El problema se ha abordado de forma tradicional mediante la extracción de una serie de características del objeto modelo, en el primer *frame* de la secuencia, que han de ser lo más robustas posibles a variaciones en posición, translación, rotación, escala incluso cambios de forma y color.

Todas estas transformaciones que puede sufrir el *target* a lo largo de la secuencia, así como lo comentado anteriormente sobre la generalidad requerida por el sistema hacen del seguimiento de objetos una de las tareas más desafiantes en el campo de la visión artificial.

2.1.1 Categorías

Según la duración de los videos y el objetivo del *tracker* podemos clasificar los sistemas de *tracking* en dos categorías fundamentales, los orientados a búsqueda a corto plazo (**short-term**) [11] y los orientados a búsqueda a largos plazo (**long-term**) [12]. El enfoque que se sigue en estas categorías es ligeramente distinto. La primera está formada por sistemas ligeros que se centran únicamente en el seguimiento del objeto. Suele ser sencillos y pueden contar con algún método de actualización del modelo [13] [14]. Los de la segunda categoría, en cambio, suelen disponer de varios módulos y tienen que estar preparados para diferentes situaciones, como la desaparición del objeto de la escena o a grandes cambios de apariencia de éste. Es por lo que suelen incluir opciones de redetección y actualizaciones de modelo más potentes [15]. También han de ser más robustos ante distractores, oclusiones y otros tipos de fenómenos que dificultan el seguimiento. Este trabajo está centrado sobre los del primer tipo, aunque sus aplicaciones podrían ser útiles para modificar y/o hacer *trackers long-term* más robustos ante la aparición de oclusiones parciales o distractores.

2.1.2 Sistemas de tracking tradicionales

En este trabajo entendemos los sistemas de seguimiento tradicionales como aquellos que no hacen uso de técnicas de *deep learning*, y que conformaban el estado del arte antes de la llegada de estos. Este tipo de sistemas se pueden dividir de forma genérica en las siguientes partes [16]:

- **Representación del objeto:** Si bien el modo más habitual es representar el objeto mediante un rectángulo que lo contiene (conocido como *bounding box)*, otras muchas formas de representarlo han sido utilizadas a lo largo del tiempo. Además, la forma de representar el objeto facilita la explotación posterior de las características de un tipo determinado. Algunas de las posibles representaciones son (ver Figura 1 para algunos ejemplos gráficos de estas):
    - Descripción mediante puntos de interés: útiles para representar objetos pequeños o puntos de interés concretos [17].
    - Formas geométricas básicas.
    - Contorno del objeto.
    - Modelo de esqueleto [18].
    - Densidades de probabilidad en la apariencia del objeto: Se puede representar un objeto mediante la densidad de probabilidad que tienen algunas de sus características como, por ejemplo, sus colores.
    - Basado en partes: Se divide el objeto en diferentes partes y se extrae características para cada una de ellas (ver Figura 9). Tienen la ventaja de ser más robustos frente a oclusiones [19].

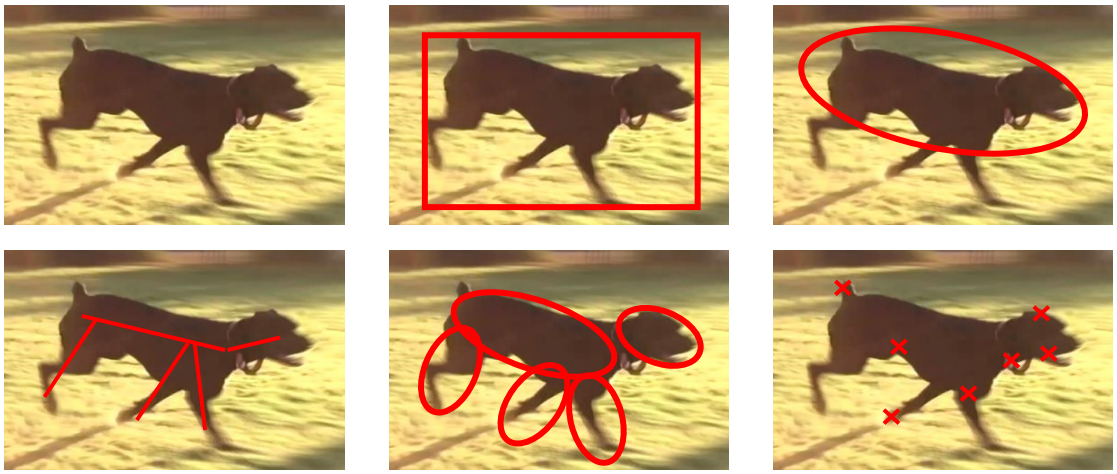


Figura 1: Ejemplos de diferentes formas de representar al objeto target. Primera fila, de izquierda a derecha: imagen original, región rectangular, región ovalada. Segunda fila, de izquierda a derecha: modelo de esqueleto, dividido en partes y por último basado en puntos de interés.

- **Selección de características:** Tras obtener una representación del objeto es necesario realizar una extracción de características que nos permita distinguirlo en futuras imágenes. Algunas de las características más empleadas en los sistemas de tracking tradicionales son las siguientes:
    - Color: uso del espacio de color como conjunto de características empleadas en la posterior detección del objeto.
    - Bordes: características más robustas a cambios de iluminación que las basadas en colores.
    - *Optical Flow* (trayectoria): se emplea para la extracción de los vectores de movimiento del objeto y de sus partes, permitiendo así deducir su posición futura y descartar zonas estáticas [20].
    - Textura: Medida de las variaciones de intensidad en una superficie. Característica más compleja que el color, pero más robusta frente a cambios.
    - Características HOG (*Histogram of Oriented Gradients*): Robustas ante oclusiones [21].

- **Detección del objeto:** Mediante el uso de las diferentes características obtenidas se busca ubicar el objeto en la siguiente imagen del video con la mayor precisión posible. Hay diferentes métodos para lograr esto, algunos ejemplos son:
    - Resta de *frames*: forma sencilla de obtener la variación entre imágenes consecutivas permitiendo así observar el desplazamiento del objeto.
    - Optical Flow: Forma más sofisticada de predecir y detectar el cambio de posición. Más costosa computacionalmente pero mucho más precisa. Permite predecir la posición futura del objeto en base a los vectores de movimiento calculados previamente [20].
    - Substracción de fondo: Eliminar el fondo para quedarnos solo con el objeto que nos interesa detectar.

- Filtros de correlación: muy populares debido a su gran rendimiento y alta precisión [22]. Se trata de buscar las características obtenidas del objeto *template* en el nuevo *frame* mediante el uso de un filtro que maximiza el valor de la correlación cruzada en la posición del objeto. Este filtro ha de ser entrenado para que se adapte adecuadamente al objeto que se desea seguir. Se explicarán en profundidad más adelante.

- **Actualización de modelo:** Este paso, si bien no está presente en todos los sistemas de seguimiento, es habitual para intentar mitigar el problema de la deriva (pequeñas variaciones en el *target* que provocan que, según avanza la secuencia cueste más al *tracker* seguirlo) comentado anteriormente. Suele realizarse mediante una media ponderada entre la representación del objeto obtenida en el primer *frame* y las nuevas características obtenidas en el *frame* actual. El problema que esto presenta es que en caso de una mala predicción la información con la que se actualizará el modelo será incorrecta provocando una degradación del modelo original. Para evitar esto muchos sistemas solo realizan la actualización en caso de que la nueva predicción supere un umbral de confianza previamente establecido como parámetro del sistema [13] [14] [23].

2.1.3 Sistemas basados en técnicas de aprendizaje profundo

En los últimos años el campo de la visión artificial ha cambiado completamente con la rápida evolución de las redes neuronales y los sistemas de *deep learning*. Estos sistemas se están empleando en el estado del arte de prácticamente todo el ámbito y el tracking no es una excepción. Numerosas aproximaciones diferentes se han realizado consiguiendo en general grandes resultados y desbancando en prácticamente todos los casos a los sistemas tradicionales.

Como se puede observar en competiciones internacionales como el Visual Object Tracking Challenge (VOT) [12] cada año es mayor el número de soluciones basadas en redes neuronales que se presentan, frente al resto de sistemas que no paran de descender. El problema a la hora de aplicar este tipo de implementaciones a sistemas de *tracking* es el rendimiento. En el seguimiento por vídeo suele ser condición necesaria el procesamiento de la información en tiempo real puesto que muchas de sus aplicaciones son sobre cámaras (ya sean cámaras de vigilancia o de un coche autónomo, por poner un par de ejemplos ya comentados). Por lo tanto, no se puede hacer uso de redes neuronales tan grandes y pesadas como en otras áreas como pueden ser la detección de objetos o la segmentación. A raíz de esto han surgido soluciones que, haciendo uso de redes relativamente pequeñas, consiguen explotar las características de los objetos sin por ello renegar de la eficiencia, consiguiendo un buen *frame rate* a la vez que una buena precisión. En la siguiente sección haremos un repaso, de forma cronológica, a los diferentes sistemas que han conformado el estado del arte y han dado forma al actual.

2.1.3.1 Filtros de Correlación

Ya hemos comentado muy brevemente algo sobre estos sistemas en el apartado 2.1.2, pero en esta sección lo haremos en mayor profundidad. Para ello nos centraremos en el sistema [22] presentado en 2010 que, si bien no es el primero, es el que mostró la potencia y versatilidad de estos en los sistemas de *tracking*. Este sistema no hace uso de técnicas de *deep learning* pero su relevancia es tal en algunos de los sistemas pertenecientes al estado del arte actual que lo

comentaremos en esta sección. Los filtros por correlación permiten seguir objetos complejos que sufren de rotaciones, oclusiones, y otros tipos de fenómenos que dificultan la tarea de seguimiento.

En esta publicación los autores proponen un nuevo tipo de filtro de correlación basado en **el Mínimum Output Sum of Squared Error (MOSSE)** que produce filtros estables aun cuando se inicializan sobre un único *frame*. Es robusto a variaciones de luz, escala, pose y deformaciones no rígidas. Además, destaca por su rendimiento. Es capaz de "detectar" oclusiones mediante el método de *peak-to-sidelobe ratio* (que mide la amplitud del pico de la correlación comparado con su entorno). Cuando una oclusión se detecta el seguimiento se pausa y se reanuda cuando el objeto reaparece. La apariencia del *target* se modela mediante filtros de correlación adaptativos y el seguimiento se realiza mediante la convolución de las características y el filtro.

Como explicación de las diferencias entre este tipo de filtro respecto de filtros de correlación más simples (como puede ser el simple recorte de la imagen objetivo) podemos destacar que estos segundos responden fuertemente al fondo y no son robustos a cambios de apariencia. En cambio, los filtros más avanzados como el MOSSE son fuertemente discriminativos entre el frente y el fondo obteniendo así una gran precisión en proporción al coste computacional.

Esta alta capacidad discriminatoria es una de las características principales de este sistema, aun siendo un sistema extremadamente ligero, alcanzaba niveles de precisión a la altura del estado del arte en la fecha de su publicación. En la Figura 2 se puede observar la respuesta a la correlación cruzada según el tipo de filtro de correlación obtenido observándose claramente como la respuesta mejora con su uso. Este equilibrio entre potencia y rendimiento hizo que los filtros de correlación se volviesen extremadamente populares entre los diferentes sistemas de tracking. Muestra de ello son los resultados de varios VOT (Video Object Tracking) Challenge consecutivos [10].

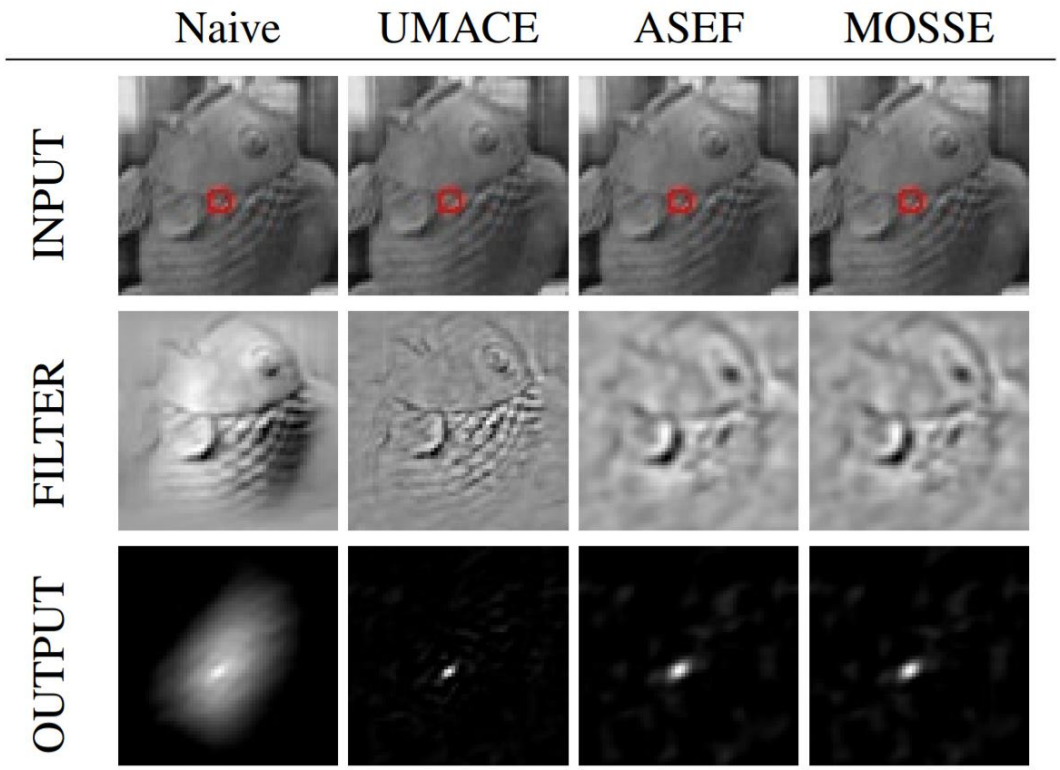


Figura 2: Respuesta a la correlación de diferentes tipos de filtros de correlación. Imagen adaptada de [22]

El funcionamiento del sistema se podría resumir de la siguiente forma:

- Los *trackers* basados en filtros de correlación como este modelan la apariencia del objeto usando el filtro, el cual ha de ser entrenado con un conjunto de imágenes. Para entrenar este filtro en concreto su utiliza la imagen del objeto obtenido del *frame* original y sobre ella se realizan varias transformaciones afines para generar un conjunto de 8 imágenes del objeto *target* cada una con diferentes perturbaciones.
- Tras obtener el filtro ya entrenado, el *target* se localiza en el siguiente *frame* mediante la correlación entre de la nueva imagen con este. Tras obtener la nueva ubicación el filtro se actualiza.
- La correlación se realiza en el dominio de la frecuencia, previo cálculo de la transformada rápida de Fourier (FFT). Para evitar artefactos en los bordes se aplica una ventana coseno a la imagen, dando así mayor peso a las muestras más cercanas al centro.

2.1.3.2 Redes neuronales Siamesas

Entrando ya en el terreno del *deep learning* toca ahora hablar de la arquitectura que constituye el estado del arte en la actualidad en sistemas de tracking *short-term* desde hace años: las redes neuronales siamesas.

Este tipo de redes se utilizan para medir la similitud entre dos pares de muestras o imágenes [24]. Se caracterizan por tener dos ramas (*branches*) que comparten los mismos pesos. A cada rama se le introduce una de las imágenes que se desean comparar. Las dos ramas se unen al final dando así una medida de similitud entre las imágenes que se han introducido en cada rama. Básicamente, las ramas se utilizan para extraer un vector de características por cada imagen (al ser los pesos de las ramas compartidos se extraerán las mismas características para cada imagen de forma que estas son comparables) y la unión entre las dos ramas se utiliza para realizar la comparación entre estas características. Esta comparación nos permite saber si ambas imágenes son equivalentes o no. El significado de esta equivalencia dependerá del tipo de entrenamiento de la red neuronal. Antes de entrar a comentar diferentes arquitecturas siamesas específicas para *tracking*, vamos a explicar uno de los primeros usos que estas tuvieron, el reconocimiento *one-shot,* ya que será útil para entender el funcionamiento de los sistemas que conforman el estado del arte actualmente.

2.1.3.3 One shot Recognition

Podemos definir el reconocimiento *one-shot* como la tarea por la cual se clasifican numerosas muestras futuras a partir de una única muestra de la clase objetivo. Se podría explicar como un problema de medida de la similitud entre diferentes muestras. Para ello, se suele usar una función que mide la distancia o diferencia entre ellas, pudiendo distinguir según esta si las muestras pertenecen a la misma clase o no. Este es un problema clásico del reconocimiento facial donde, en muchas ocasiones, se dispone de una única imagen de la cara de individuo y a partir de ella se han de clasificar otra serie de imágenes como pertenecientes, o no, a esta misma persona. [24]

Este problema suele resolverse mediante el uso de redes neuronales siamesas. De estas redes se obtienen unos *embeddings* (tensores de características obtenidos por la red y que representan a la imagen de entrada) por cada imagen que son comparables (puesto que representan el mismo conjunto de características). La idea es que estos son muy similares en

caso de que la imagen pertenezca a la misma persona y muy distintos en caso de que pertenezca a personas diferentes. Para maximizar la eficiencia de este tipo de redes se suelen usar en entrenamiento funciones como la *triplet loss* [25].

La función de perdidas conocida como *triplet loss* se caracteriza por el uso de grupos de tres imágenes durante el entrenamiento de redes *one-shot*. La imagen objetivo, una imagen de la misma clase y una imagen de una clase diferente. Se han de seleccionar las imágenes positiva y negativa adecuadamente en orden de maximizar el entrenamiento todo lo posible (si las imágenes seleccionadas como negativas son muy distintas a las positivas la red no aprenderá a diferenciar entre clases similares lo que provoca una bajada de rendimiento en la fase de test). Matemáticamente esta se puede definir mediante la Ecuación 1.

$$L(A,P,M) = \max\left(\left||f(A) - f(P)\right||^2 - \left||f(A) - f(M)\right||^2 + \alpha, 0\right) \quad \textit{(Ecuación 1)}$$

Donde A es la imagen objetivo, P es una imagen de la misma clase que A, M es una imagen de una clase distinta, α es un término conocido como margen y que se utiliza para evitar que la red converja en la solución arbitraria $f(A) = f(P)$ para cualquier imagen. La función $f(\cdot)$ representa el *embedding o vector de características*.

Este tipo de aproximación es interesante puesto que es la que utilizan algunos de los sistemas de *tracking* que conforman el estado del arte actual [8][9][10]. La *triplet loss* también ha sido utilizada en el entramiento de diferentes sistemas de *tracking* [25].

2.1.3.4 Fully-Convolutional Siamese Networks for Object Tracking (SiamFC)

Ahora sí, empezamos con los sistemas de *deep learning* puramente dedicados al seguimiento de objetos. La primera arquitectura que comentaremos será la que sentó las bases del estado del arte y en la que se basan fundamentalmente la mayoría de *trackers short-term* actuales; la red SiamFC [26].

Esta red siamesa extrae los *embeddings* tanto de la imagen objetivo (el modelo obtenido en el primer *frame*) como de la zona de búsqueda en la siguiente imagen, definida como la posición del objeto en el *frame* anterior más un cierto margen. Tras obtener las características de cada imagen, simplemente se aplica la correlación entre los *embeddings* del objeto y los de la zona de búsqueda. La posición espacial para la cual se obtiene el máximo de la correlación se considera la nueva ubicación del objeto. Una imagen de la arquitectura empleada en este sistema puede encontrarse en la Figura 3.

De este sistema destaca su simplicidad (no hace actualización de modelo ni se mantiene ningún tipo de memoria del modelo) consiguiendo un equilibrio entre eficiencia (funciona hasta a unos 86 *frames* por segundo o fps) y eficacia.

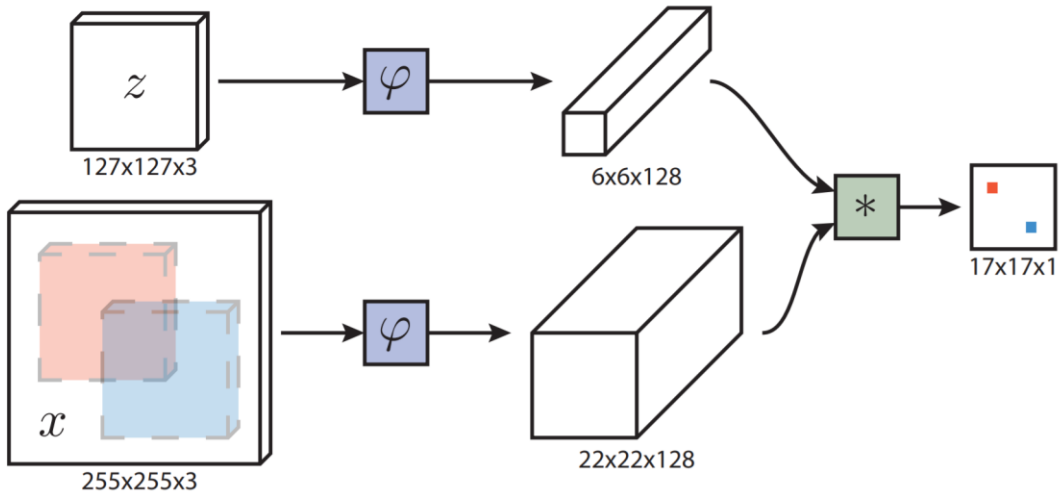


Figura 3: Arquitectura de la red SiamFc. Imagen adaptada de [26]

2.1.3.5 End-to-end representation learning for Correlation Filter based Tracking (CFNet)

CFNet [1] es una variación de la arquitectura anterior. Como SiamFC utiliza una red neuronal siamesa con arquitectura AlexNet como *backbone* (la red neuronal que compone cada rama) pero cambia en cómo se unen ambas ramas. Mientras que en el caso anterior se realiza una correlación cruzada de ambos mapas de características, en esta se calcula un filtro de correlación, similar a los vistos anteriormente [22], a partir de los *embeddings* obtenidos del objeto *target*. Además, al ser parte de la red neuronal, puede ser entrenado junto al resto de la red de forma conjunta o *end-to-end*. Este nuevo bloque se puede entender como la creación de un *template* discriminativo que es robusto a translaciones.

Es interesante recalcar que, como se ve en la Figura 4, esta arquitectura aun haciendo uso de redes siamesas es asimétrica. Además, a diferencia de en [26] el modelo se actualiza mediante una media móvil con la información aportada por cada nuevo *frame* y hace uso de un proceso multiescala para adaptarse adecuadamente a los cambios de tamaño.

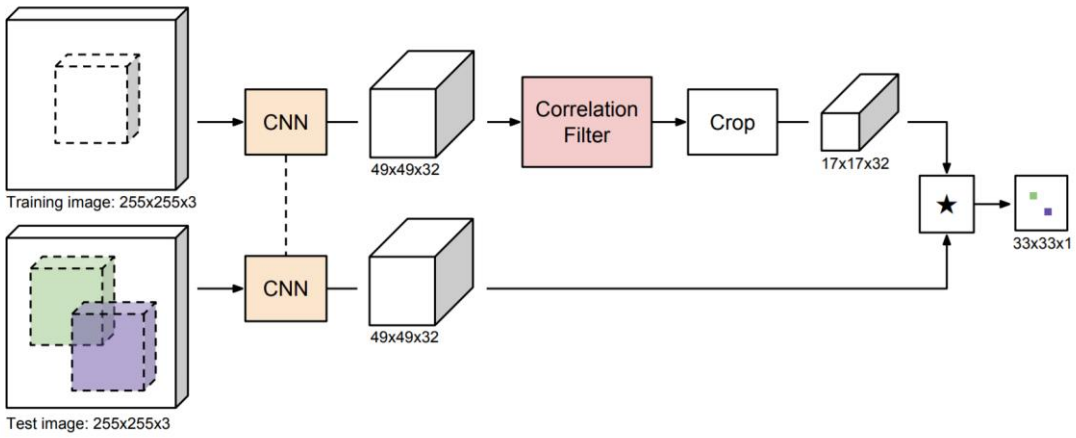


Figura 4: Arquitectura de la red CFNet. Imagen adaptada de [1]

2.1.3.6 High Performance Visual Tracking With Siamese Proposal Network (SiamRPN)

SiamRPN [27] también hace uso de redes neuronales siamesas a la hora realizar el seguimiento, pero, a diferencia de la anteriores, hace uso de una subred de propuesta de regiones de interés (RPN) como la empleada en Faster R-CNN [28].

Esta subred RPN es la encargada de presentar diferentes candidatos a objeto dentro de la imagen, así como su posición. Hace uso de *anchors* para definir los distintos tamaños y relaciones de aspecto de los *bounding-box* que emplea. Por cada *bounding-box* que encuentra realiza dos tareas:

- Utiliza una rama de clasificación para identificar si esa región corresponde a frente o a fondo de la imagen, es decir si contiene o no algún objeto de interés.
- Utiliza una segunda rama de regresión para devolver las coordenadas que tendría ese *bounding-box* dentro de la imagen original. Hay que tener en cuenta que esta subred trabaja con mapas de características de tamaños completamente distintos a los de la imagen original, es por ello por lo que este paso es necesario para obtener la posición adecuada.

Además, hay que destacar que RPN es capaz de extraer posibles candidatos más rápido y con más precisión que sistemas tradicionales como la búsqueda selectiva o como los empleados en otros sistemas como Fast R-CNN [29].

Este sistema se podría resumir como una red compuesta de dos partes, como se ve en la Figura 5, una subred siamesa para la extracción de características y una subred RPN que contiene ramas de clasificación y de regresión que da las posibles localizaciones del objetivo.

En la **fase de inferencia** (o localización), que se puede ver en la Figura 6, la red se formula como un problema de **identificación one-shot**. Se computan los *kernels* del objeto a partir del primer *frame* usando la RPN y solo con la información obtenida de este primer *frame* se realiza el seguimiento en el resto de la secuencia. Sobre el mapa de características obtenido y tras la convolución con el *kernel* se obtiene la posición del objeto mediante un sistema de selección de las diferentes propuestas generadas.

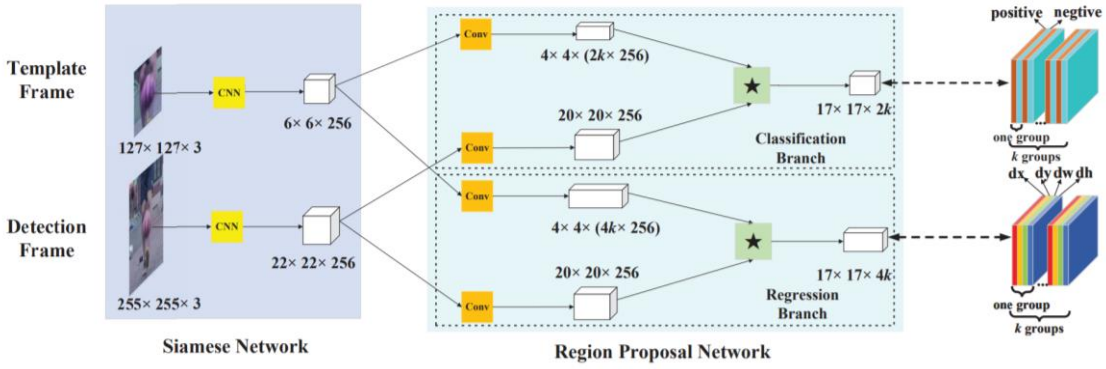


Figura 5: Arquitectura de la red SiamRPN en entrenamiento. Imagen adaptada de [27]

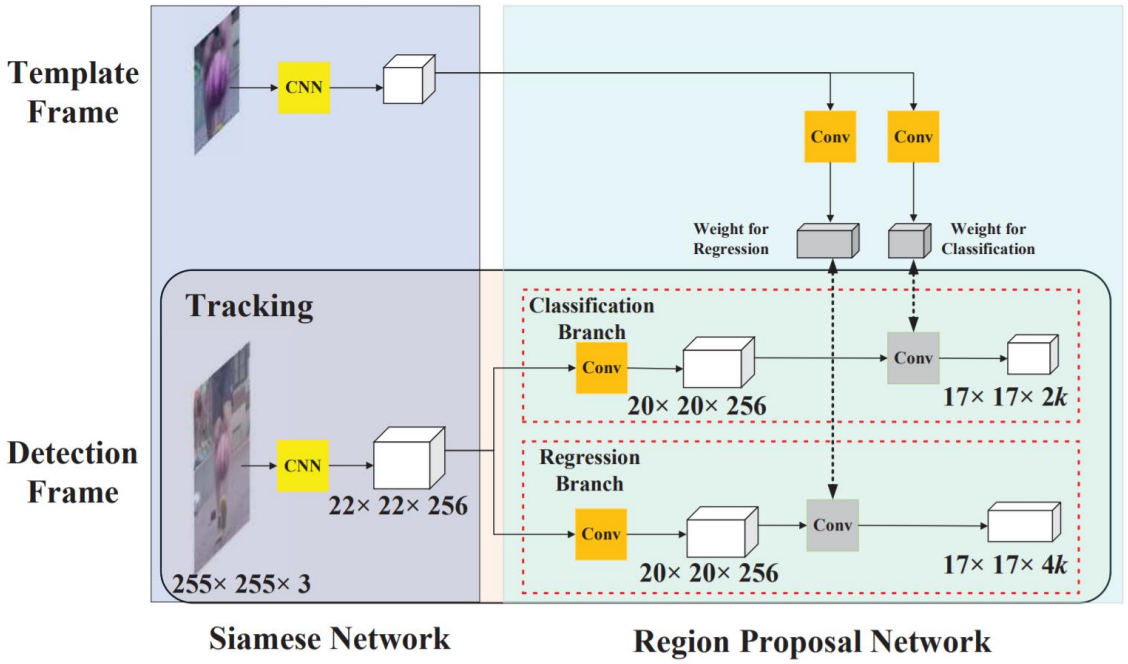


Figura 6: Arquitectura de la red SiamRPN en fase de inferencia. Imagen adaptada de [27]

Este sistema no utiliza ni test multiescala (gracias a la subred RPN) ni actualizaciones de modelo, lo que permite que funcione hasta a 160 fps. Todo el sistema se entrena de forma conjunta (*end-to-end*).

2.1.3.7 SiamRPN++: Evolution of Siamese Visual Tracking with Very Deep Networks

La arquitectura presentada en [3] es la evolución de la red anterior. Sus principales diferencias radican en la red utilizada como *backbone* de la arquitectura siamesa (que pasa de ser AlexNet a ResNet50). Esto, que parece un cambio trivial, lleva asociado en este artículo un estudio en profundidad sobre las causas de por qué el uso de redes muy profundas como ResNet provocan, en sistemas de tracking similares a este, resultados peores que al usar redes más pequeñas, así como una explicación de las medidas que se pueden tomar para evitar esto y así aprovechar la potencia de estas redes.

Los autores concluyen que esta incompatibilidad entre las redes muy profundas y los *tracker* siameses es debido a la necesidad de que estos últimos sean invariantes a translaciones lo cual solo se puede cumplir en redes sin *padding*. El problema es que el *padding* es inevitable en las redes muy profundas para mantener una resolución suficiente a la entrada de la red que resista los diezmados (*poolings*) sucesivos de las redes profundas, lo cual rompe la restricción de invariancia a translaciones.

Otra de las aportaciones de este trabajo es la demostración de que la agregación de los *embeddings* obtenidos en diferentes capas permiten al *tracker* obtener más información, puesto que capas de diferentes niveles contienen niveles de abstracción de información distintos.

2.2 PRINCIPALES LIMITACIONES

Tras comentar algunos de los diferentes sistemas que conforman el estado del arte actual procederemos ahora a comentar algunos de los problemas típicos de la tarea de seguimiento de objetos. Y es que ya hemos hablado en varias ocasiones de los diferentes fenómenos que se pueden dar durante el tracking y que hacen más difícil el seguimiento correcto del objeto. Así que, en este apartado, haremos un repaso de los diferentes problemas, así como sus implicaciones. Algunos ejemplos de estos se pueden ver en la Figura 7. Destacamos los siguientes:

- Oclusiones: Se pueden dividir en parciales y totales. Pueden provocar la pérdida del objeto con facilidad y, en el caso de las oclusiones totales, se requiere de un módulo de redetección para poder recuperar el objeto. Otro problema es la actualización de modelo sobre un objeto parcialmente ocluido. Esto da lugar a una alteración de las características del objeto dificultando así el seguimiento posterior [4].
- Cambios de iluminación: Provocan cambios repentinos en las características del objeto (locales) o de la escena entera (globales).
- Distractores: se definen como la presencia de objetos muy similares al que se está siguiendo dentro de la escena, lo cual dificulta la tarea de seguimiento [2].
- Movimientos rápidos: la mayoría de los *trackers* realizan la búsqueda del objeto en una región cercana a su última posición conocida. Si el objeto se mueve demasiado deprisa corremos el riesgo de que salga de esta zona de búsqueda y, por lo tanto, sea imposible de encontrar. Aumentar la zona de búsqueda es una posible solución, pero conlleva un incremento del coste computacional, encontrar un equilibrio entre ambos factores es indispensable.
- Cambios de apariencia: Similar a los cambios de iluminación. Según cambian las características del objeto en el trascurso de la secuencia más difícil es seguirlo. Modelos de actualización como los ya comentados ayudan a paliar esto, pero aun así no pueden hacer nada frente a cambios repentinos.

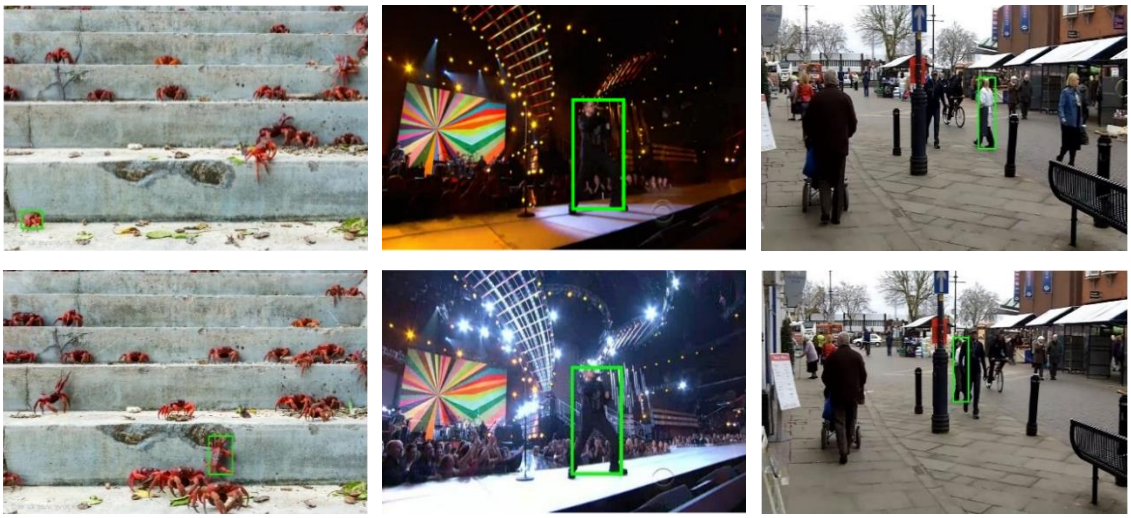


Figura 7: Ejemplos de diferentes situaciones que pueden complicar el seguimiento del objeto. Por columnas de izquierda a derecha: presencia de distractores, cambios de iluminación en la escena y oclusiones.

En este trabajo nos hemos centrado en desarrollar un sistema que ayude a mitigar los problemas derivados de la presencia de oclusiones parciales o distractores en la escena. Este no es ni mucho menos un planteamiento nuevo y, como veremos a continuación, numerosas aproximaciones diferentes se han seguido a lo largo del tiempo para intentar paliar los efectos de estos problemas. A continuación, comentaremos algunos sistemas de seguimientos orientados explícitamente a ser robustos frente al problema de la oclusión.

2.2.1.1 Sistemas de detección de oclusiones

En [13] se propone un sistema basado en un *kernel* de estructura circular que, como en otros muchos *trackers*, se aplica en el dominio frecuencial. Lo característico de este sistema es la inclusión de un sistema para detectar oclusiones, así como el uso de un procesado diferente según si el objeto se encuentra ocluido o no.

Este sistema guarda la representación del objeto en cada *frame* en un *buffer*. Para discriminar entre oclusiones utiliza dos medidas de distancia. La primera se obtiene haciendo una "distancia de oclusión" entre el *patch* (o región) objetivo con los *patch* del entorno. Esto se basa en que si el objeto es ocluido lo será por uno de los objetos de su alrededor y, por lo tanto, la distancia bajará (el objeto ahora será muy similar al entorno). La segunda distancia es la distancia entre el *patch* que representa al objeto en el *frame* actual y los *patches* que representan al objeto en los *frames* anteriores que se encuentran almacenados en el buffer. Ambas distancias se miden usando la Local HOG Distance (LHD). En caso de detectar oclusión usando la combinación de ambas distancias la actualización de modelo se interrumpe y se selecciona un modelo más representativo de entre los almacenados en el *buffer*.

El mayor inconveniente de este sistema es que todo este procesamiento extra lleva un coste computacional asociado que influye negativamente en la eficiencia del sistema.

En [30] podemos encontrar otra de las alternativas propuestas para lidiar con el problema de las oclusiones. En este trabajo se hace uso de dos cámaras para así obtener información de profundidad similar a la visión humana permitiendo así una reconstrucción de la escena en 3D y detectar las oclusiones. Esto obviamente no es viable en la mayoría de los casos ya que la mayor parte de los videos son captados con una única cámara.

En los sistemas basados en filtros por correlación tradicionales [22] se utiliza una técnica simple para detectar la presencia de oclusiones e intentar mitigar los problemas al actualizar el modelo. Esta detección se realiza observando el resultado de la correlación. Las oclusiones parciales dan lugar a un PSR (*peak to sidelobe ratio*) bajo, así como a valores en el mapa de correlación bajos. Un ejemplo de esto se puede observar en la Figura 8.

Por último, hay que destacar los sistemas basados en partes [14]. Este tipo de sistema subdivide el objeto en zonas cada una con su propia representación. Esto los hace muy robustos ante oclusiones parciales puesto que es suficiente con que alguna de las partes del objeto no se encuentre ocluida para poder detectarlo. El problema de este tipo de sistemas es el coste computacional asociado a todo ese procesado. Un ejemplo de esta división se puede observar en la Figura 9.

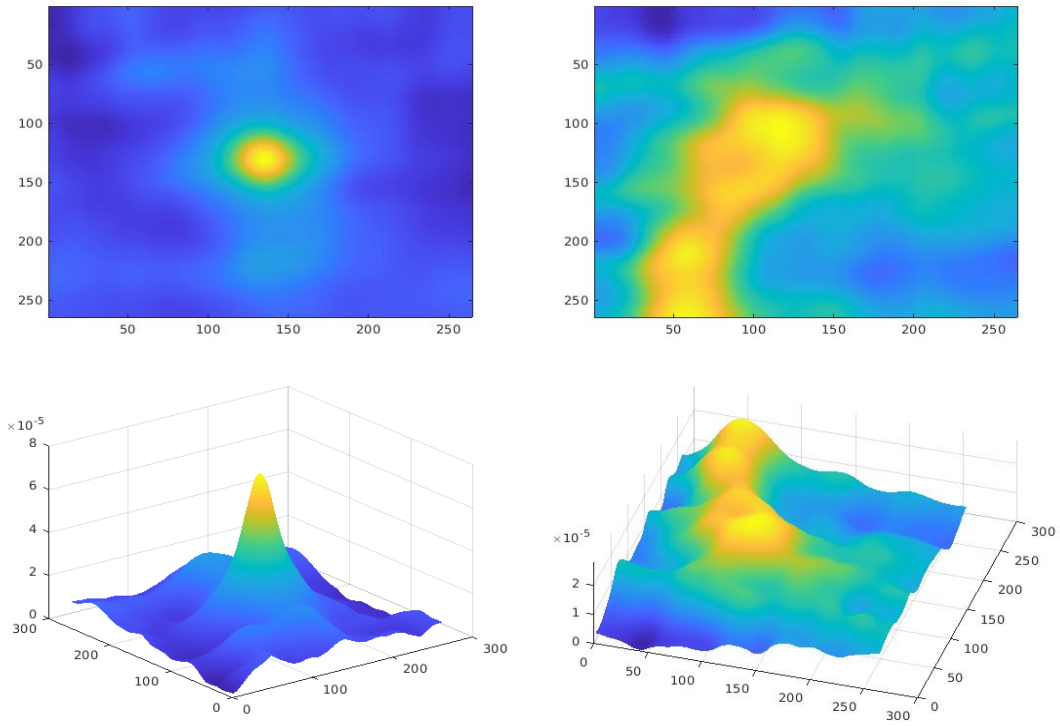


Figura 8: Ejemplos PSR. La columna de la derecha muestra un mapa de correlación con una PSR alta mientras que la de la derecha muestra un mapa con una PSR baja.

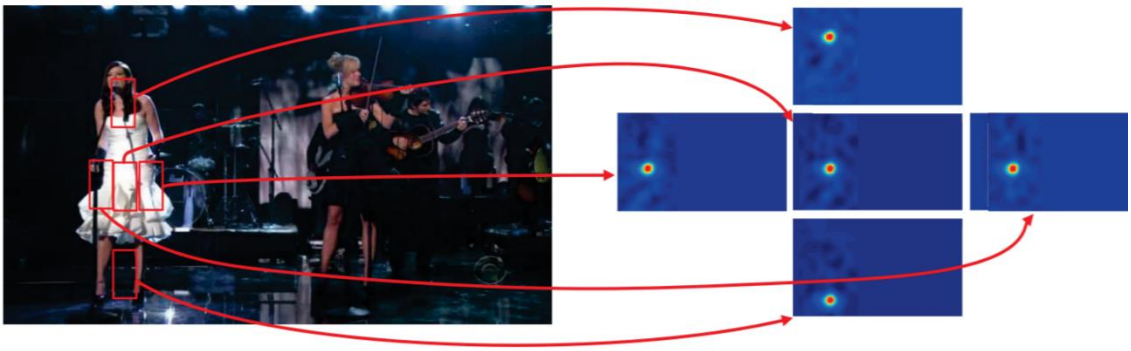


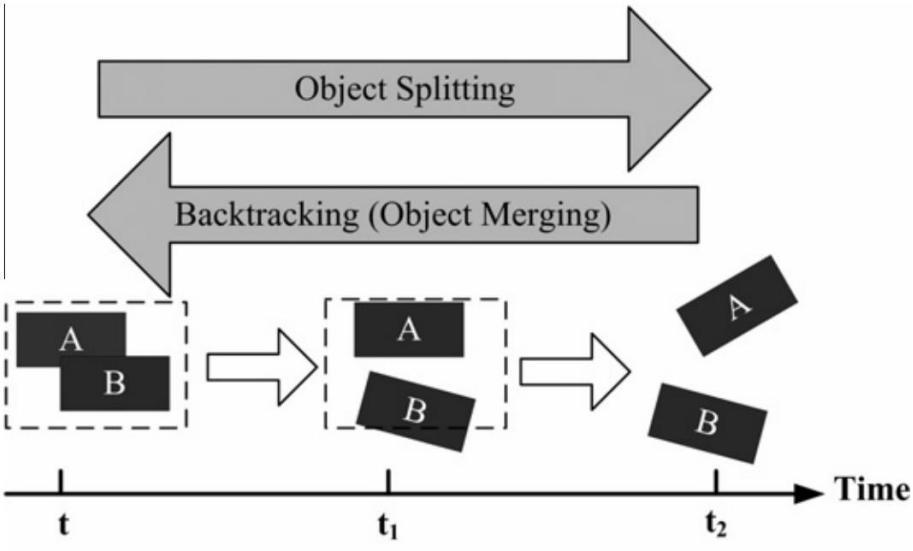
Figura 9: Ejemplo de sistema basado en partes. A la hora de hacer el seguimiento del target este se descompone en partes y se hace el seguimiento de cada parte de forma individual. Imagen adaptada de [31]

2.2.1.2 Sistemas que hacen uso de Backtracking

Otra técnica empleada ocasionalmente para detectar posibles oclusiones ha sido la conocida la conocida como *backtracking* o seguimiento hacia atrás. Podemos entenderla como el proceso por el cual un *tracker* realiza el seguimiento sobre el objeto *target* del video en la dirección contraria a la habitual, desde el *frame* actual hasta un *frame* anterior. Los resultados de este proceso pueden ser utilizados por el *tracker* para diferentes cosas. Por ejemplo, en [31] se hace uso de este método para detectar oclusiones. En esta publicación se presenta un sistema,

basado en partículas, con el objetivo de seguir múltiples objetos, dirigido a aplicaciones en detección de eventos. Lo que nos interesa de este sistema es su método para gestionar las posibles oclusiones en los objetos al inicio del video mediante el uso del *backtracking*. En los casos en lo que se detecta una posible oclusión inicial (algo que se consideraba un único objeto cuando se inicializó el *tracker* se divide en dos) se realiza el *backtracking* por cada "parte" en la que se ha dividido y se comprueba si al realizar esto el *tracker* converge en el objeto original. En caso de que así sea se considera que ambos objetos estaban ocluidos inicialmente y se considera cada una de las partes un objeto único a seguir. En la Figura 10 se puede ver un diagrama que ejemplifica el funcionamiento de este sistema.

En [32] este método se emplea dentro de un sistema basado en filtros de correlación como los vistos en [22]. El *backtracking* se realiza junto al PSR con el objetivo de detectar oclusiones y, en caso de detección, desactivar la actualización de modelo. Para ello se propone realizar un seguimiento hacia atrás del objeto de un único paso, del *frame* actual al *frame* anterior. Tras este proceso se calcular el IoU (Intersection over Union, ver sección 4.1) obtenido y combinándolo con el PSR resultante del mapa, se decide si el *target* esta ocluido o no



.

Figura 10: Sistema empleado en [32] para detectar posibles objetos ocluidos al inicio de la secuencia mediante backtracking. Imagen adaptada de [32]

2.3 DATASETS

Con el cambio de paradigma en el campo del *computer vision* y el paso al uso de sistemas de *deep learning* la cantidad de datos requeridos para entrenar estos sistemas ha aumentado de manera exponencial. La aparición de numerosas bases de datos de imágenes etiquetadas responde a esta necesidad y permite un entrenamiento offline más robusto mediante el uso de millones de imágenes. De hecho, los *tracker* actuales que representan el estado del arte son aquellos que mediante el uso de una aproximación *one-shot* permiten un entrenamiento en base a pares de imágenes que no requieren ser ni del mismo video [27]. Esta potencia adicional permite entrenar estos sistemas usando bases de datos que no tienen por qué estar realmente adaptadas para tracking, si no a otras tareas como el reconocimiento de objetos o incluso a la segmentación de escenas. Mientras el *bounding-box* del objeto esté etiquetado estas imágenes será adecuadas para entrenar la red. A continuación, se comentarán algunos de los datasets utilizados en este trabajo:

- **VOT Dataset:** La tendencia actual en la construcción de muchos *datasets* es la de incrementar el número de secuencias todo lo posible sin prestar demasiada atención a la calidad de las anotaciones y el contenido. En algunos casos se pueden encontrar mezcladas imágenes en blanco y negro de un solo canal con otras de los tres canales RGB y, en la mayoría de los casos, no se anotan los diferentes atributos del video, como oclusiones o cambios de iluminación a nivel de *frame*, lo cual ha dado numerosos problemas en este trabajo a la hora de intentar encontrar imágenes con anotaciones adecuadas para su realización [19]. En cambio, en este *dataset* podemos encontrar una cantidad pequeña de videos, pero con una gran variedad de situaciones y con diferentes atributos anotados por cada imagen.
- **Need for Speed Dataset (NFS)**: Base de datos consistente en 100 videos capturados a un gran frame rate (240 fps) con oclusiones anotadas *frame* a *frame* [33].
- **NUS PRO Dataset:** Contiene 365 secuencias con anotaciones manuales de *bounding-box* y oclusiones [34].

Una comparativa entre los diferentes *datasets* puede encontrarse en la Tabla 1.

Dataset	*Nº videos*	*Nº frames*	*Oclusiones*	*Distractores*
VOT	60	18.000	✓	✓
NFS	100	380.000	✓	✓
NUS	365	109.500	✓	✓

Tabla 1: Comparativa de las características de los diferentes datasets.

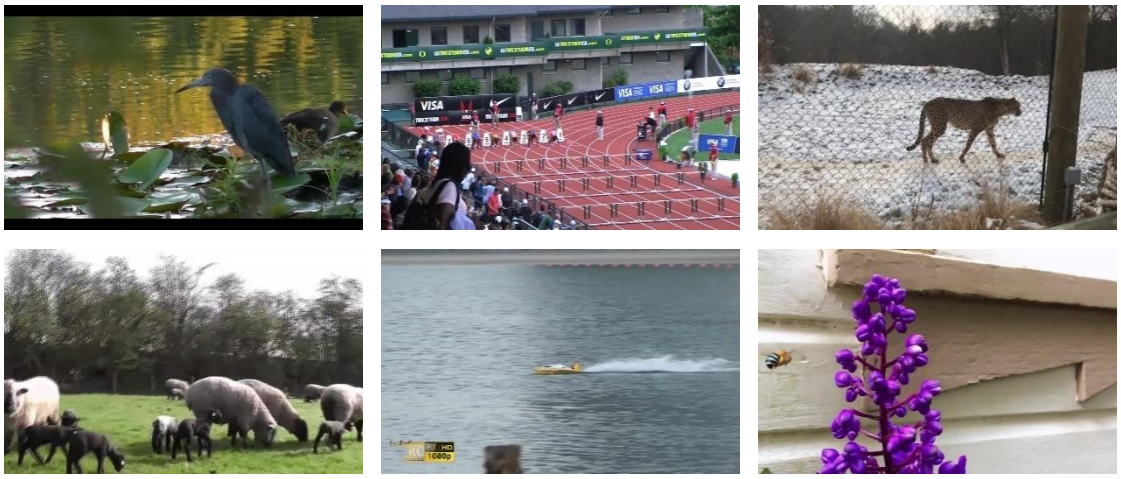


Figura 11: Algunas secuencias procedentes de los diferentes datasets: Columna izquierda VOT dataset, columna central NUS dataset y columna derecha NFS dataset

2.3.1.1 Limitaciones de los dataset existentes

Uno de los problemas encontrados en la mayoría de los *dataset* con los que se ha trabajado en este proyecto ha sido la ausencia de oclusiones etiquetadas a nivel de *frame* las cuales son útiles a la hora de validar el sistema y realizar pruebas. Los pocos que incluyen estas anotaciones las incluyen de manera imprecisa y sin seguir criterios unificados como, por ejemplo, que tanto por cierto de la imagen ha de estar ocluida para ser considerada como tal. Por todo ello se planteará la creación de un nuevo *dataset,* generado a partir de los ya comentados en este apartado, con nuevas etiquetas de oclusión más robustas.

Otro de los problemas que encontrados ha sido la practica ausencia de oclusiones en la mayoría de los videos. Muchas bases de datos contienen cientos de video, pero solo un pequeño porcentaje de estos contiene alguna oclusión. Además, todas las oclusiones que se dan dentro de un mismo video suelen ser debidas a un único objeto lo que provoca una falta de variedad en los datos y que limita en cierta medida la validación del sistema frente a oclusiones, sobre todo a nivel de *frame*. Simplificando mucho se podría decir que cada video que contiene oclusiones aporta fundamentalmente una única muestra nueva al *dataset*.

Para intentar subsanar este problema se planteó inicialmente un sistema generador de oclusiones artificiales mediante *alpha bleding* como forma de aumentar la variedad de la base de datos y poder comprobar de forma individual el comportamiento de las diferentes implementaciones realizadas en este trabajo frente a oclusiones de diversos tipos, pero finalmente no ha sido empleado en este trabajo. Más información sobre este sistema puede encontrarse en el Anexo A.

3 DISEÑO Y DESARROLLO

3.1 INTRODUCCIÓN

En esta sección se describen los diferentes sistemas implementados en este trabajo, así como su función y posibles aplicaciones. En el apartado 2 hemos comentado el funcionamiento de varios *trackers* y su posible división en dos grandes categorías, *short-term* y *long-term* [11] [15]. Esta división no es algo casual y se debe principalmente a la incapacidad de los *trackers* puros a seguir el objeto *target* durante un periodo de tiempo largo. Esto es debido en gran parte a la aparición de los problemas que hemos comentado anteriormente (apartado 2.2) y que a la larga provocan perdidas en la efectividad del *tracker*.

Es por ello por lo que en este trabajo nos hemos querido centrar en formas de mitigar y hacer más robustos los sistemas de seguimiento ante situaciones de este tipo, más concretamente frente a oclusiones y distractores. Para ello se han desarrollado diferentes sistemas con este objetivo empleando la técnica, comentada anteriormente, conocida como *backtracking* o seguimiento hacia atrás.

Generalmente los diferentes sistemas de seguimiento que implementan soluciones ante estos problemas, como los vistos en la sección 2.2.1.1 , siguen un planteamiento sencillo. Por ejemplo, muchos de los sistemas a la hora de gestionar las oclusiones realizan una detección simple, basándose en el PSR y en caso de que esta confianza baje simplemente desactivan la actualización de modelo del objeto que se está siguiendo [22]. Esto tiene la ventaja de que es computacionalmente muy ligero, pero a cambio no permite reaccionar ante situaciones más complejas, como la presencia de distractores o a la pérdida del *target*.

En cambio, en este trabajo se ha seguido un planteamiento diferente. Nuestros sistemas no buscan la detección de oclusiones o distractores para desactivar la actualización y continuar con el procesado, sino que, siguiendo un enfoque más completo, en caso de detectarse pérdida de confianza de la red en el objeto seguido, ya sea debido a oclusiones parciales, cambios de apariencia o distractores, se realiza un proceso de refinado de candidato. En este proceso, nuestro sistema decide una serie de potenciales candidatos que podrían representar al objeto seguido en este *frame* y a cada uno de ellos, mediante el uso de *backtracking*, se le otorga una puntuación. El candidato con mayor puntuación es elegido nueva posición del objeto en ese *frame*. Un ejemplo de este funcionamiento sobre distractores se puede ver en la Figura 12.

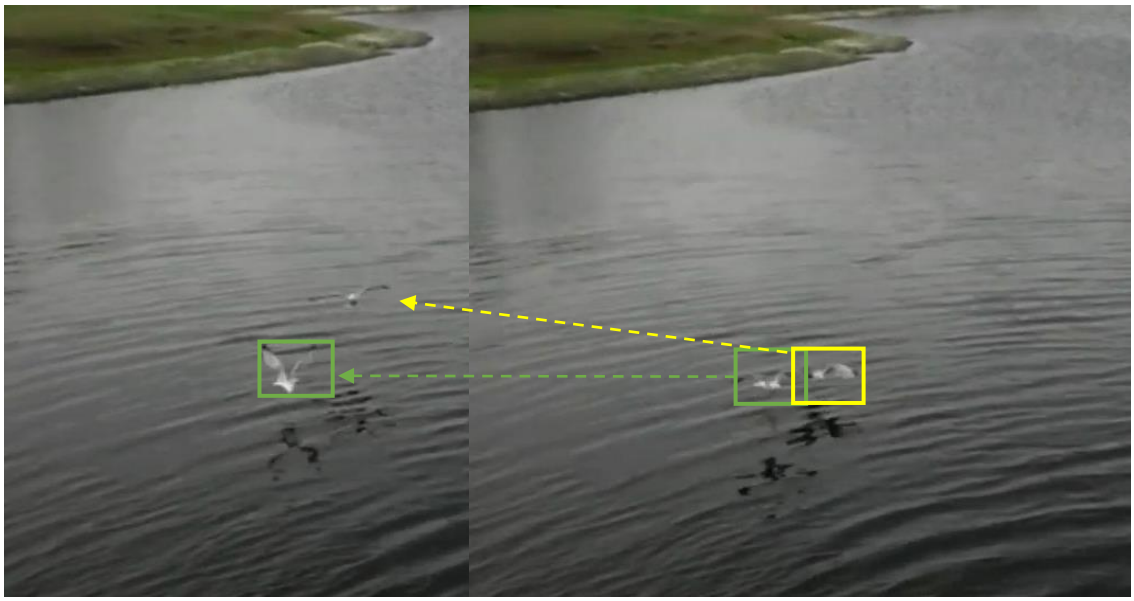


Figura 12: Ejemplo de funcionamiento de backtracking sobre distractores. La primera imagen representa un frame del video en el instante t y la segunda imagen un frame en el instante $t+N$. En esa segunda imagen aparecen dos objetos (gaviotas) prácticamente idénticas una al lado de la otra, en verde la que deseamos seguir y en amarillo la que actúa como distractor. Para permitir distinguir entre ambas nuestro sistema sigue la trayectoria marcha atrás de cada una (del frame $t+N$ hasta el frame t), representadas mediante las flechas, y a partir de la ubicación obtenida para cada gaviota en el frame t y de la información del tracking en ese frame, representada mediante el rectángulo verde, podemos desambiguar la correcta.

Este sistema de refinamiento de candidatos debería dar resultados mejores frente a oclusiones parciales o distractores debido al uso del *backtracking*. En el caso de que varios candidatos similares (presencia de distractores) se encuentren en el área de búsqueda, al realizar el seguimiento hacia atrás de cada uno veremos cómo solo el candidato correcto nos lleva hacia las posiciones en las que habíamos detectado el objeto en los *frames* anteriores, mientras que el candidato del distractor al realizar el seguimiento hacia atrás llevará a la posición del distractor en estas imágenes previas que, normalmente, será distinta a la obtenida por nuestro *tracker*. Esta idea está ejemplificada en la Figura 15.

Ante la presencia de oclusiones parciales la lógica es algo diferente. En estos casos la confianza de los resultados dados por la red suele bajar en gran medida, dando lugar a mapas de correlación con numerosos máximos (Figura 13), es decir, varios candidatos potenciales. Al realizar el seguimiento hacia atrás de cada candidato solo el candidato adecuado debería converger en las posiciones que hemos obtenido para el objeto al realizar el seguimiento mientras que el resto de los candidatos convergerán hacia posiciones diferentes, permitiendo así desambiguar correctamente la posición correcta del candidato pese a la oclusión. Uno de los potenciales problemas en estos casos es que, si la oclusión es completa o prácticamente completa, las características del objeto pueden verse tan distorsionadas que el proceso de *backtracking* no converja para ninguno de los posibles candidatos.

Figura 13: Resultados obtenidos por la red tracker sobre el objeto target (la chica vestida de blanco) sin ocluir y sobre el objeto target parcialmente ocluido. Las zonas amarillas son las que tienen una mayor puntuación. Se puede ver como en el caso del objeto ocluido la confianza de la red baja habiendo una zona de incertidumbre (amarillo-verde) en la que podría estar el objeto.

En este trabajo se han propuesto dos aproximaciones diferentes, cada una con diferentes variantes. Una primera, basada en un ***backtracking*** más **simple**, que solo se utiliza en caso de detección de pérdida de confianza en la salida de la red, y en el que para cada posible candidato se realiza un seguimiento de N*frames* hacia atrás con el objetivo de comprobar donde converge. Se selecciona el candidato cuya trayectoria obtenida sea la más similar a la trayectoria obtenida durante el seguimiento normal.

La segunda aproximación también hace uso del ***backtracking*** pero, en este caso, se realiza para todos los *frames* del video pero únicamente un *frame* hacia atrás, del *frame* actual al *frame* inmediatamente anterior. Las puntuaciones obtenidas por cada candidato se van **acumulando** y cada nueva puntuación tiene en cuenta las puntuaciones obtenidas en los *frames* anteriores. De esta forma se va generando un tipo de árbol de caminos cada uno con una puntuación acumulada. Siguiendo este árbol cada candidato obtendrá una puntuación y se seleccionará aquel con mayor.

Nuestros sistemas se han integrado junto a la red vista anteriormente SiamFC [26]. Además, el sistema de **backtracking simple** también se ha integrado y validado sobre la red CFNet [1]. La forma en que se integran estos sistemas puede verse en la Figura 14. Haciendo uso tanto de las características (*embeddings*) obtenidos por estos *trackers* como del propio sistema de seguimiento que se ha empleado a la hora de realizar el *backtracking*. Así pues, el objetivo final es la mejora de estos *trackers* mediante el uso de nuestro sistema.

Como ya se ha explicado anteriormente, estas redes detectan la posición del nuevo objeto basándose únicamente en medidas de similitud espacial (correlación) entre el objeto *target* y la imagen en la cual se busca ubicar este objeto sin tener en cuenta nada más. Nuestro sistema se puede interpretar como una forma de añadir información temporal a este proceso puesto que

el refinamiento de candidatos al realizar el *backtracking* tiene en cuenta la trayectoria seguida por el objeto a la largo de la secuencia y actúa en consecuencia.

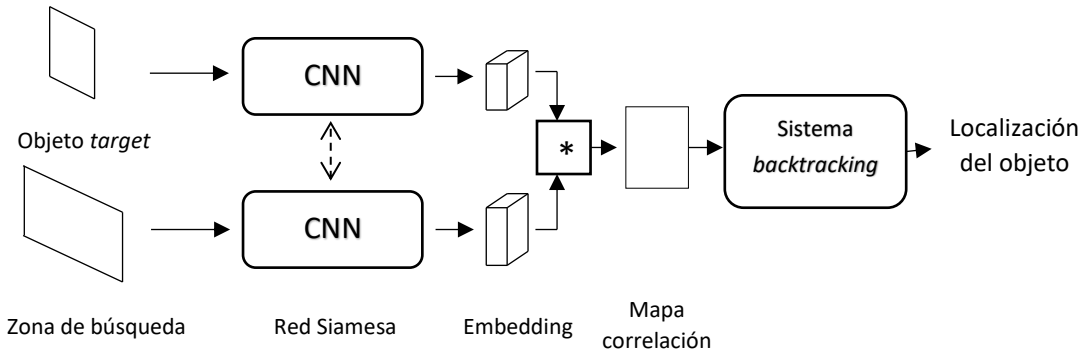


Figura 14: Sistema completo con el modelo de selección de candidatos mediante backtracking.

3.2 BACKTRACKING SIMPLE

Este primer algoritmo funciona de la siguiente manera:

- Para cada *frame* la red siamesa devuelve un mapa con los resultados de la correlación cruzada obtenida entre el objeto original y la zona de búsqueda del objeto en ese *frame*. En la red original se selecciona el máximo de este mapa como la nueva posición del objeto en ese *frame*. En el sistema propuesto este paso funciona de manera diferente. En los casos que los que el mapa de lugar a varios máximos diferentes que superan un cierto umbral (por ejemplo, el 80% del máximo absoluto del mapa), como en la Figura 15, en vez de seleccionar el máximo absoluto como nueva posición consideramos cada uno de estos máximos como un posible candidato a la nueva posición.
- Para cada uno de los candidatos se realiza el siguiente proceso: para un número de *frames* configurable N se realiza un seguimiento del candidato desde el *frame* actual t hasta el *frame* t-N (tracking inverso *o backtracking)*. A partir de los resultados obtenidos para ese candidato podemos comprobar si el camino marcha atrás seguido por este es similar al que hemos recorrido al hacer el seguimiento normal. Teóricamente si el candidato es el correcto, al hacer el seguimiento hacia atrás este nos llevará por un camino similar al que ha recorrido el objeto al seguirlo hacia adelante. En cambio, si el candidato es incorrecto el camino que recorreremos hacia atrás será diferente puesto que no estaremos siguiendo el objeto original. Un ejemplo de este comportamiento se puede observar en la Figura 15.

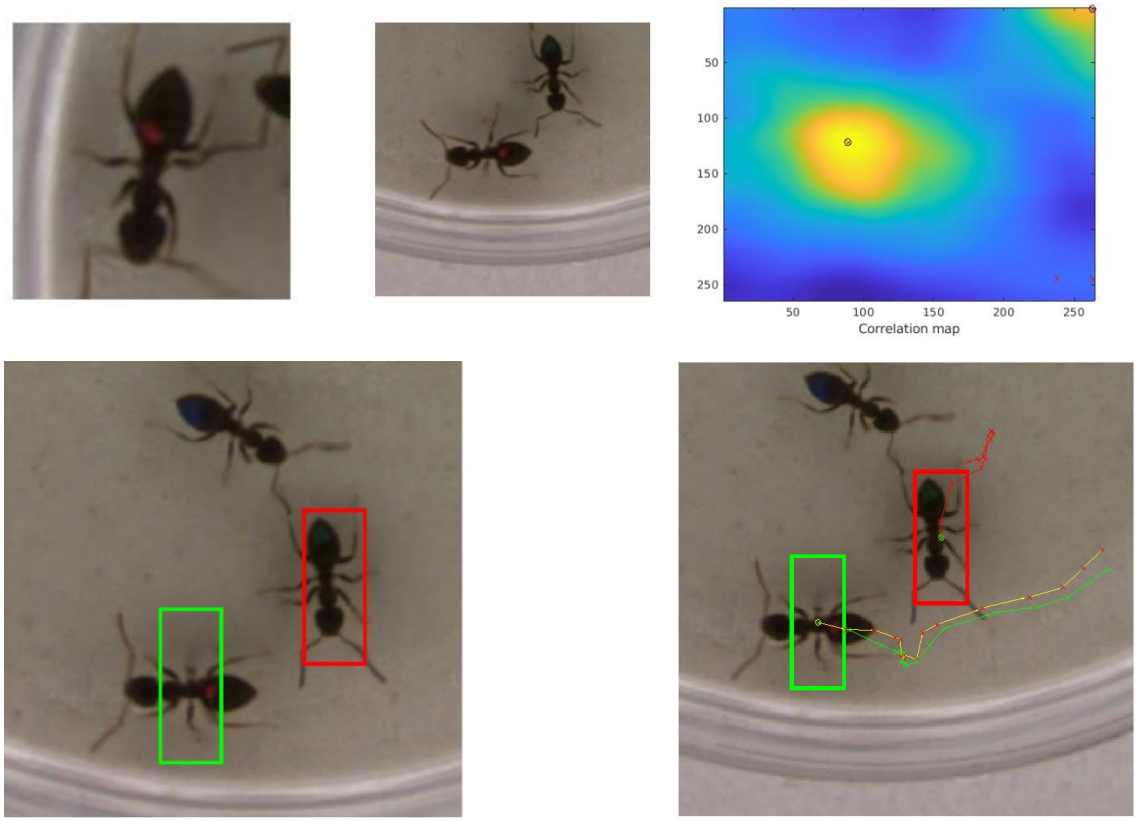


Figura 15: Ejemplo de funcionamiento del sistema de backtracking simple. Primera fila: La primera imagen es el objeto a seguir durante la secuencia (el target obtenido en el primer frame). La segunda imagen es la zona de búsqueda para un frame del video. La tercera imagen es el mapa de correlación obtenido de la aplicación del target sobre la zona de búsqueda. Como se puede ver se detectan dos máximos sobre esta imagen, uno en la zona central y otro en la esquina superior derecha. Segunda fila: La primera imagen representa los dos candidatos asociados a estos máximos. La segunda imagen representa las trayectorias obtenidas mediante backtracking para cada candidato. En verde se representa la trayectoria que se ha seguido en los frames anteriores al realizar el seguimiento normal, en rojo la trayectoria seguida por el candidato rojo al usar el backtracking. En amarillo la trayectoria seguida por el candidato target al realizar el backtracking. Como se puede observar, estas trayectorias permiten desambiguar sin problemas entre ambos candidatos al seleccionar la más similar a la original.

3.3 BACKTRACKING ACUMULADO

El otro método utilizado en este trabajo es similar al anterior, en el sentido de que utiliza *backtracking* para refinar posibles candidatos, pero sigue una aproximación algo diferente. En este caso para cada *frame* se realiza el *backtracking* de los candidatos (aunque solo haya uno posible) pero solo entre el *frame* actual y el inmediatamente anterior (de *t* a *t-1*). Según los resultados obtenidos se da una puntuación a cada candidato, pero, a diferencia de en el caso anterior, esta puntuación también tiene en cuenta los resultados anteriores, es decir de las puntuaciones obtenidas por los candidatos del *frame* anterior. De esta forma se va creando una suerte de árbol de posibles caminos combinando las puntuaciones *frame* a *frame* como se puede ver en la Figura 16. En los casos en lo que hay varios posibles candidatos se selecciona aquel cuya puntuación combinada de *backtracking* y camino más próximo sea mayor.

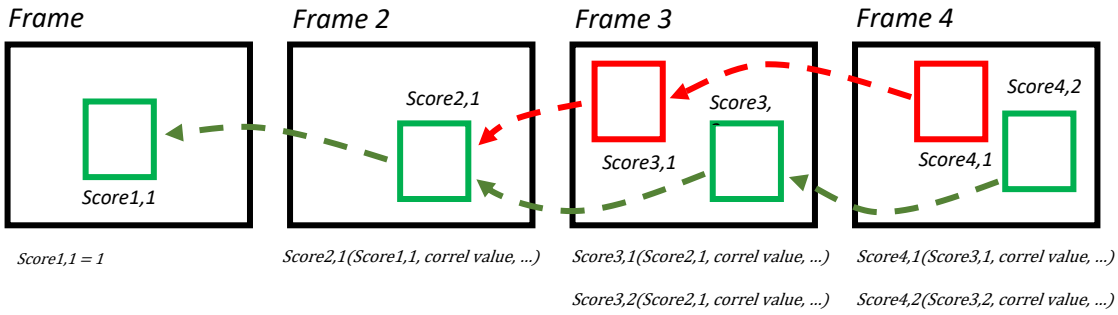


Figura 16: Ejemplo de creación de árbol de caminos a partir de los diferentes candidatos obtenidos en cada uno de los cuatro primeros frames. El score de cada candidato calculado tiene en cuenta el valor del candidato obtenido en el frame anterior, por ejemplo, el Score3,1 (frame 3, primer candidato) depende del score previo Score2,1 (frame 2, primer candidato)

Formalmente este algoritmo para un *frame t* se puede definir como:

i. Se obtienen los diferentes máximos del mapa de correlaciones producido por la red neuronal y se seleccionan los que superen un cierto umbral. Cada uno de estos se considerará un candidato.

ii. Para cada uno de estos candidatos:
- Se realiza un único paso de *backtracking*, del *frame t* al *frame t-1* y se almacena el *bounding-box* obtenido.
- Se calcula el solape de este *bounding-box* con los de los candidatos detectados en el *frame* anterior. Se asigna el candidato actual a un único candidato previo: aquel que resulte en mayor solape (medido mediante la IoU) tras el *backtracking*.
- Se calcula la puntuación del candidato combinando medidas propias del candidato en este *frame* (valor del máximo en el mapa de correlación, IoU obtenido al hacer el *backtracking* con el candidato previo asignado, …) con la puntuación obtenida por el candidato previo que procede del *frame* anterior. De esta forma se va teniendo en cuenta el camino previo recorrido al combinar las puntuaciones de los candidatos con los procedentes del *frame* anterior.

iii. Se selecciona como nueva posición del objeto el candidato con mayor puntuación.

Se guardan las puntuaciones y posiciones de **todos** los candidatos detectados en este *frame* para que puedan ser empleados en el *frame* siguiente. Un ejemplo básico de funcionamiento puede verse sobre la Figura 17.

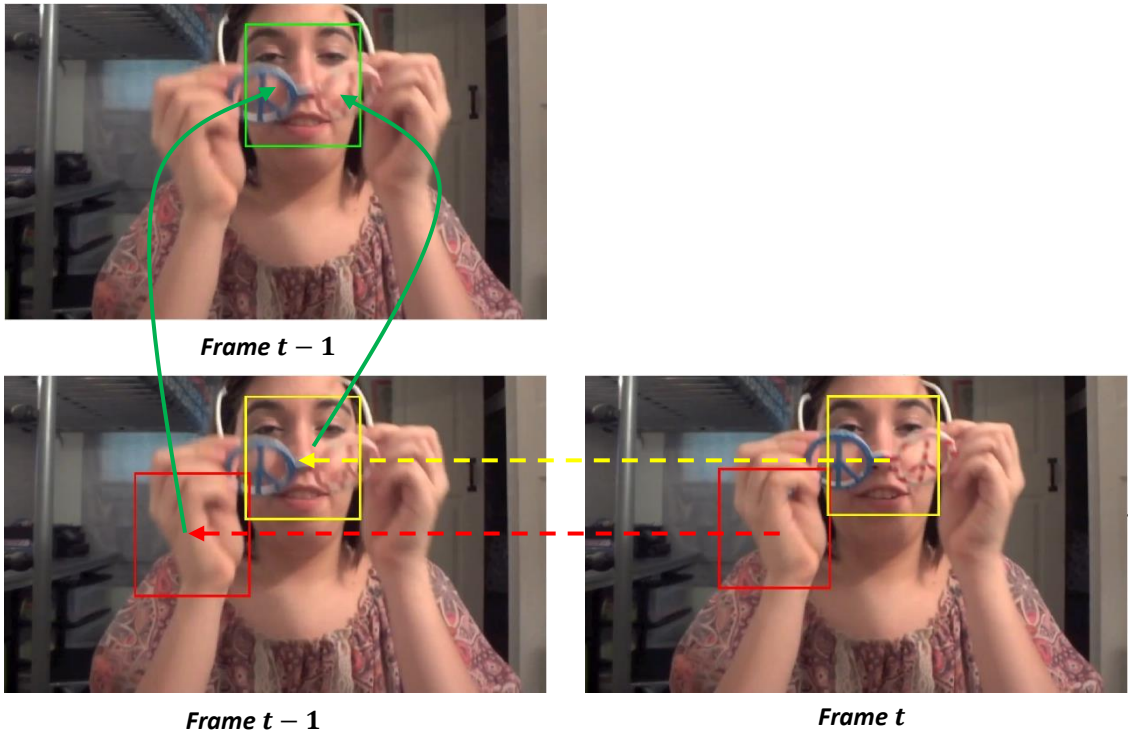


Figura 17: Ejemplo de funcionamiento del backtracker acumulado. La columna de la izquierda es el frame anterior y la de la derecha el frame actual. Primero se detectan los candidatos en el frame actual (en este caso dos). Se hace el backtracking del cada candidato al frame anterior. Cada posición se compara con los candidatos que se habían obtenido para el frame anterior (uno solo en este caso, representado en verde) y a partir de esta comparación se obtiene una puntuación para cada candidato. El candidato con mayor puntuación es seleccionado.

Uno de los problemas de este planteamiento es lo que hemos llamado la presencia de *cuellos de botella*. Si por cualquier caso, para un *frame* determinado, el número de candidatos baja mucho o el candidato ideal (el objeto *target*) no es detectado, ya sea por su salida de escena, debido a una oclusión total o a un cambio de aspecto repentino, tendremos el problema de que el camino que nos debería llevar al objeto original no es recuperable. Un ejemplo de esto se puede ver en la Figura 18. Uno forma propuesta para mitigar este problema es la propagación de candidatos previos que no son detectados en los *frames* sucesivos, que se puede observar en la Figura 19 y cuyo planteamiento es el siguiente:

- Si para un *frame* t se detecta y escoge un candidato, pero en el *frame* posterior no se detecta ningún candidato en esa misma zona, este candidato anterior se propaga al *frame* siguiente como si se hubiese detectado un máximo en ese lugar.

Con esta salvedad, el resto del algoritmo se comporta de forma idéntica. Este planteamiento permite sobre todo en caso de oclusión total no perder la información de la última posición del objeto ni la puntuación acumulada que este tenía, aunque esta se ve penalizada en cierto grado.

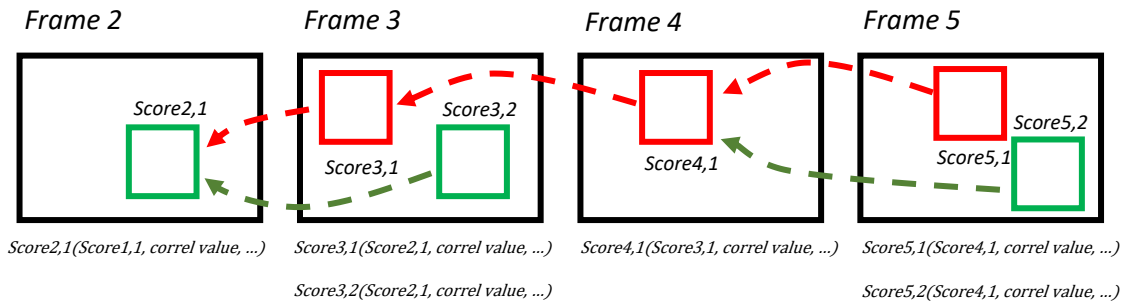


Figura 18: Ejemplo de cuello de botella en frame 4. En ese frame el candidato target no es detectado lo que provoca una pérdida de su puntuación acumulada. Por ello, aunque en el frame 5 se vuelve a detectar, se ve perjudicado al realizar el proceso de backtracking puesto que en el frame 4 no hay ningún candidato al que converger durante el backtracking.

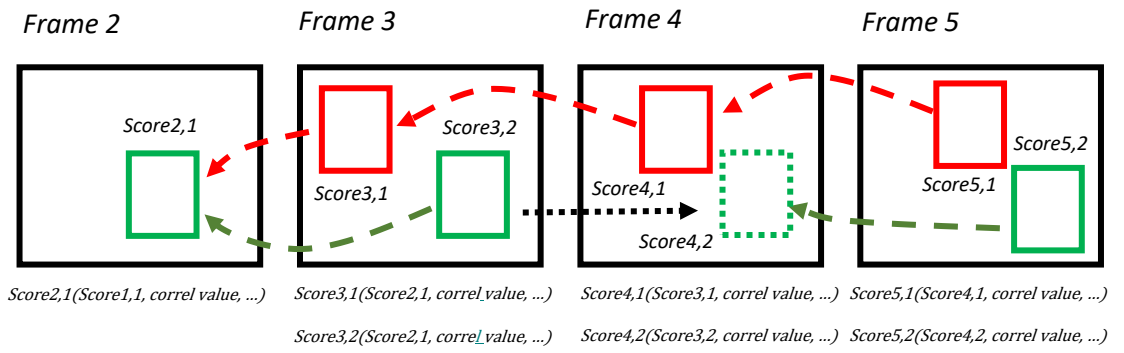


Figura 19: Ejemplo de propagación de candidato (flecha negra) para evitar el cuello de botella en el frame 4. De esta forma el candidato correcto en el frame 5 no pierde completamente el camino acumulado ni la puntuación.

3.4 BASE DE DATOS DE OCLUSIONES A NIVEL DE *FRAME*

Como se ha comentado en la sección 2.3, uno de los problemas que se ha encontrado para poder validar los resultados de estos sistemas, es la ausencia de bases de datos con las oclusiones correctamente etiquetadas a nivel de *frame*.

Esto ha creado la necesidad de etiquetar de nuevo una gran cantidad de secuencias, cuando no de etiquetarlas desde cero, a fin de disponer de una base de datos unificada y moderadamente robusta que poder emplear a la hora de validar los diferentes sistemas. Esta anotación nos permitirá obtener los resultados discriminando no solo sobre secuencias completas si no sobre pequeños fragmentos del video con oclusiones. Esta base de datos permite evaluar con mayor resolución temporal el rendimiento del sistema en estas situaciones, aislando las situaciones de oclusión de otro tipo de problemas o situaciones inherentes al análisis sobre secuencias completas.

La base de datos está compuesta de un total de 266 videos diferentes procedentes de los diferentes conjuntos de datos descritos en la sección 2.3. Las etiquetas utilizadas han sido las siguientes:

- *Frame* con objeto no ocluido: Etiqueta 0

- *Frame* con objeto parcialmente ocluido: Etiqueta 1
- *Frame* con objeto totalmente ocluido: Etiqueta 2

Algunos ejemplos del uso de estas etiquetas sobre varias secuencias se pueden ver en la Figura 20.

Dado que no todos los conjuntos de datos presentan situaciones de oclusión, la variedad del conjunto de datos construido no es todo lo grande que se desearía, pero lo consideramos un buen punto de partida para la evaluación segmentada de las situaciones de oclusión.

Figura 20: Ejemplo de videos con sus oclusiones etiquetadas a nivel de frame. En verde el ground truth si el objeto no está ocluido, en amarillo si está ocluido parcialmente y en negro si está completamente ocluido.

4 PRUEBAS Y RESULTADOS

En este trabajo hemos realiza las diferentes pruebas sobre un subconjunto de videos formado a partir de los datasets comentados anteriormente. Conforman un total de 266 videos con un total de 77548 *frames.* Antes de presentar los diferentes resultados obtenidos en ambos sistemas pasaremos a comentar las diferentes métricas de evaluación que se emplean para evaluar los resultados de los *trackers*.

4.1 MEDIDAS DE RENDIMIENTO

A la hora de analizar el funcionamiento de un sistema de seguimiento de objetos es importante tener claro las diferentes medidas empleadas en el estado del arte para medir el rendimiento de los diferentes sistemas y así poder compararlos adecuadamente. Si bien hay numerosas formas de cuantificarlo (desde métodos clásicos como la precisión, el *recall* o el *f-score* hasta métodos como la curva-ROC) los más empleados en los diversos *challenges* [12] son los siguientes:

- **Accuracy**: Mide el grado de solape entre el *bounding-box* predicho por el *tracker* con el *ground truth* (posición real del objeto). Se suele conocer como IoU (Intersection over Union) y su fórmula viene dada por la Ecuación 2 [35].

$$accuracy(A, B) = \frac{A \cap B}{A \cup B}$$

(Ecuación 2)

  Donde A es el rectángulo que contiene la posición estimada del objeto (este rectángulo se suele denominar *bounding-box*) y B es el rectángulo que contiene la posición autentica del objeto (conocido esta posición como el *ground-truth*)
- **Robustez**: número de fallos (perdidas del *target*) por secuencia. Se considera fallo cuando el IoU con respecto al *ground-truth* cae por debajo de un cierto umbral.
- **EAO** (expected average overlap): Utilizado inicialmente en el *VOT challenge* [12] se ha convertido *de facto* en una de las medidas de rendimiento más empleadas. Mide el solapamiento del *tracker* en caso de que no se reinicie al perder el objetivo. Es una mezcla del *accuracy* y el número de pérdidas del *target*.
- **FPS**: frames per second. Mide la eficiencia del sistema. La mayor parte de los sistemas que conforman el estado del arte actual destacan por conseguir unos muy buenos resultados en este campo permitiendo el procesamiento de los *frames* en tiempo real, con una velocidad de procesamiento de hasta 100 fps.

Otra característica que destacar a la hora de medir el rendimiento en el seguimiento de objetos es la existencia de un problema conocido como deriva o *drift* que se da debido a los pequeños cambios de apariencia en el modelo original que se producen según avanza la secuencia. Esto provoca que, en videos largos, el error se acumule hasta llegar un punto en el que el *tracker* es incapaz de seguir al objeto y nunca llegue a recuperarse. Para evitar penalizar a los *tracker* completamente por esto a la hora de hacer medidas de rendimiento, se suele implementar un sistema por el cual en caso de pérdida del objetivo (o fallo), el *tracker* se reinicia a partir de un *frame* determinado, usando como nuevo modelo la posición *ground-truth* del objeto target en

ese *frame*. De esta forma se evita considerar todos los *frames* posteriores a la perdida como fallos, lo que podría dar lugar a medidas de rendimiento sesgadas.

4.2 CONFIGURACIÓN DEL SISTEMA

Tras comentar las diferentes medidas de rendimiento pasamos ahora a explicar la configuración final de los dos sistemas implementados en este trabajo, así como sus resultados. Estos resultados se desglosarán por categorías de videos, resultados en videos con presencia de oclusiones completas, resultados en videos con presencia de oclusiones parciales y resultados en videos con presencia de distractores. También se presentarán resultados a nivel de *frame,* según si estos están ocluidos o no.

4.2.1 Configuración *backtracking* simple

Este sistema tiene dos parámetros a ajustar el umbral a partir del cual un máximo en el mapa de correlación se considera un candidato (que llamaremos THR) y el número de *frames* que se realizara el *backtracking* para cada candidato (que llamaremos N). En la figura X se puede ver como varían los resultados sobre el *dataset* para diferentes Ns y THRs. Además, el score obtenido por un candidato tras realizar el proceso de *backtracking* queda definido como la Ecuación 3.

$$Score_{candidato} = \frac{1}{N}\sum_{f=1}^{N} IOU(bb_{backtracking}(f), bb_{original}(f)) \quad \textit{(Ecuación 3)}$$

Donde N es el número de *frames* sobre los que se ha realizado el *backtracking*, $bb_{backtracking}(f)$ es el bounding box obtenido durante el *backtracking* en el *frame* f y $bb_{original}(f)$ es el bounding box obtenido durante el seguimiento normal para ese *frame* e IoU la operación definida en 4.1 como *accuracy*.

De esta forma a mayor coincidencia entre la trayectoria obtenido por el candidato con la trayectoria seguida originalmente mayor será la puntuación obtenida. Como parámetros del sistema se seleccionó una $N = 10$ para evitar penalizar demasiado la eficiencia del sistema y se realizó un barrido de umbrales de detección de candidato obteniendo resultados solo sobre las partes ocluidas de las diferentes secuencias (mediante el uso del *dataset* de oclusiones generado en este trabajo) obteniendo así los resultados de la Tabla 2. A partir de estos seleccionamos el umbral $THR = 75\%$

Valor de THR	*IoU Frames Ocluidos*	*FPS*
Tracker Original	28.57	32.0
95%	29.93	17.0
90%	29.64	12.7
85%	28.49	11.0
80%	30.15	9.5
75%	30.22	7.9
70%	29.00	6.0

Tabla 2: Resultados obtenidos sobre los frames etiquetados como oclusiones en el dataset generado para este trabajo.

4.2.2 Configuración *backtracking* acumulado

En este sistema el parámetro más importante es el cálculo de la puntuación de cada candidato, ya que depende, como se ha explicado anteriormente, tanto de los candidatos anteriores como del actual. Para fusionar de manera adecuada esta información la puntuación para un candidato tras calcular su *backtracking* al *frame* anterior queda definida como en la Ecuación 4.

$$Score_{candidato} = (corr + iou_{candidatoPrevio}) * Score_{candidatoPrevio} \quad \textit{(Ecuación 4)}$$

Donde $corr$ es el valor del mapa de correlación en la posición del candidato, $iou_{candidatoPrevio}$ es el iou entre el candidato tras hacer el backtracking al frame anterior con el candidato previo más cercano y $Score_{candidatoPrevio}$ es la puntuación de este candidato previo más cercano.

De esta forma la puntuación depende del valor del candidato en este *frame* pero se ve ponderada por la puntuación del candidato asignado en el *frame* anterior. Mediante esta puntuación se tiene en cuenta el camino más probable recorrido por cada candidato y se va actualizando en cada nuevo *frame*.

Otro parámetro a tener en cuenta en este sistema es la penalización que se otorga a la puntuación de un candidato cuando este no es detectado en un *frame* y siguiendo el criterio explicado en la sección 3.3 se ve propagado al *frame* siguiente. En este trabajo se propone y evalúa, a partir del cálculo de la puntuación previamente explicada, simplemente asignar un IOU con el candidato previo de 0 de forma que la formula quedaría como se puede ver en la Ecuación 5.

$$Score_{candidato} = corr * Score_{candidatoPrevio} \quad \textit{(Ecuación 5)}$$

4.3 RESULTADOS GENERALES

Los resultados de las siguientes tablas se presentan de la siguiente forma: la medida conocida como distancia representa la distancia euclídea media entre los centros de las posiciones estimadas y las posiciones reales, el *overlap*, o IoU, es la medida presentada en la sección 4.1 y los *fps*, o *frames* por segundo, son la medida de la velocidad media a la que el sistema procesa los *frames* de una secuencia.

En los resultados expuestos en la Tabla 3 y la Tabla 4 se puede observar cómo ambos sistemas mejoran los resultados obtenidos de manera general. Como complemento a esta métrica general, se puede entender la distribución de las mejoras en relación con los videos atendiendo al grafico representado en la primera imagen de la Figura 21. En cada una de las imágenes de esta figura están representadas las mejoras en IoU en cada uno de los videos respecto de los resultados obtenidos por el *tracker* original (con estas mejoras ordenadas de menor a mayor, siendo un valor negativo un resultado peor en ese video que el del *tracker* sin módulo). En esta figura puede observarse cómo hay algunos videos que mejoran en gran medida (parte derecha de la figura) y otros que empeoran (parte izquierda de la figura). También se puede observar cómo ambos métodos tienen un gran impacto en la velocidad del sistema. El *backtracking* simple funciona a aproximadamente un cuarto de la velocidad del sistema original y el acumulado entorno a un 7.5% la velocidad del original. Otro detalle a destacar es que las mejoras obtenidas sobre la segunda red, CFNet, son mucho menores que las obtenidas sobre la red SiamFC. Para poder entender mejor estos resultados y comprobar realmente como están funcionando estos sistemas consideramos necesario desglosar los resultados en grupos según las características de los diferentes videos, es decir, según si estos contienen distractores, oclusiones parciales u oclusiones completas.

SIAM FC	*Distancia*	*Overlap (IOU)*	*FPS*
Default	60.81	36.88	33.1
Backtracking Simple	**58.02**	**38.30**	7.9
Backtracking Acumulado	58.64	38.28	2.4

Tabla 3: Comparativa de los resultados obtenidos sobre el conjunto de videos empleando la red SiamFC. A menor distancia y mayor IoU mejores resultados.

CFNet	*Distancia*	*Overlap (IOU)*	*FPS*
Default	58.63	38.37	32.6
Backtracking Simple	**57.62**	**38.72**	8.0

Tabla 4: Comparativa de los resultados obtenidos sobre el conjunto de videos empleando la red CFNet. A menor distancia y mayor IoU mejores resultados.

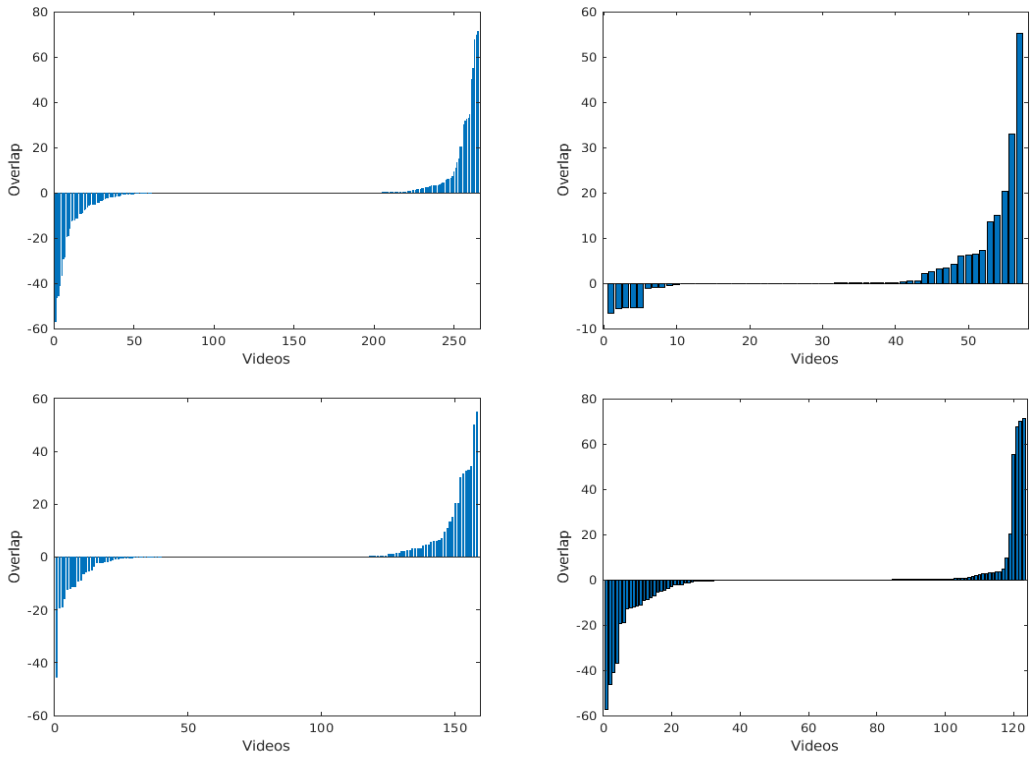


*Figura 21: Mejora en overlap (o IoU) para cada uno de los videos respecto del sistema original con los resultados ordenados de menor a mayor. Obtenido con el sistema de **backtracking simple** integrada sobre la red **SiamFc**. Resultados desglosados por categorías. Primera fila, primera imagen: resultados obtenidos para cada uno de los videos que conforman el dataset. Segunda imagen: resultados únicamente sobre videos con presencia de distractores. Segunda fila, primera imagen: resultados obtenidos sobre videos con presencia de oclusiones parciales. Segunda fila, segunda imagen: videos con presencia de oclusiones completas.*

4.4 RESULTADOS POR CATEGORÍAS DE VIDEO

4.4.1 Videos con presencia de oclusiones parciales

En la Tabla 5 y la Tabla 6 se puede observar cómo los resultados obtenidos en los videos en los que hay presencia de oclusiones parciales los resultados obtenidos mejoran en mayor proporción (mayor balance de videos en los que se mejora) que en los obtenidos en los obtenidos sobre el conjunto completo de videos, lo cual indica que nuestros sistemas se comportan mejor que la referencia en estas situaciones, pero fallan en otras. Además, mejoran ambos sistemas, tanto el simple como el acumulado. En la integración del simple sobre la red CFNet vemos como también mejora el sistema, pero en menor medida que sobre la red SiamFc.

SIAM FC	*Distancia*	*Overlap (IOU)*	*FPS*
Default	60.64	36.30	33.0
Backtracking Simple	59.07	37.34	7.9
Backtracking Acumulado	**58.79**	**37.63**	2.4

Tabla 5: Comparativa de los resultados obtenidos sobre videos con presencia de oclusiones parciales empleando la red SiamFC. A menor distancia y mayor IoU mejores resultados.

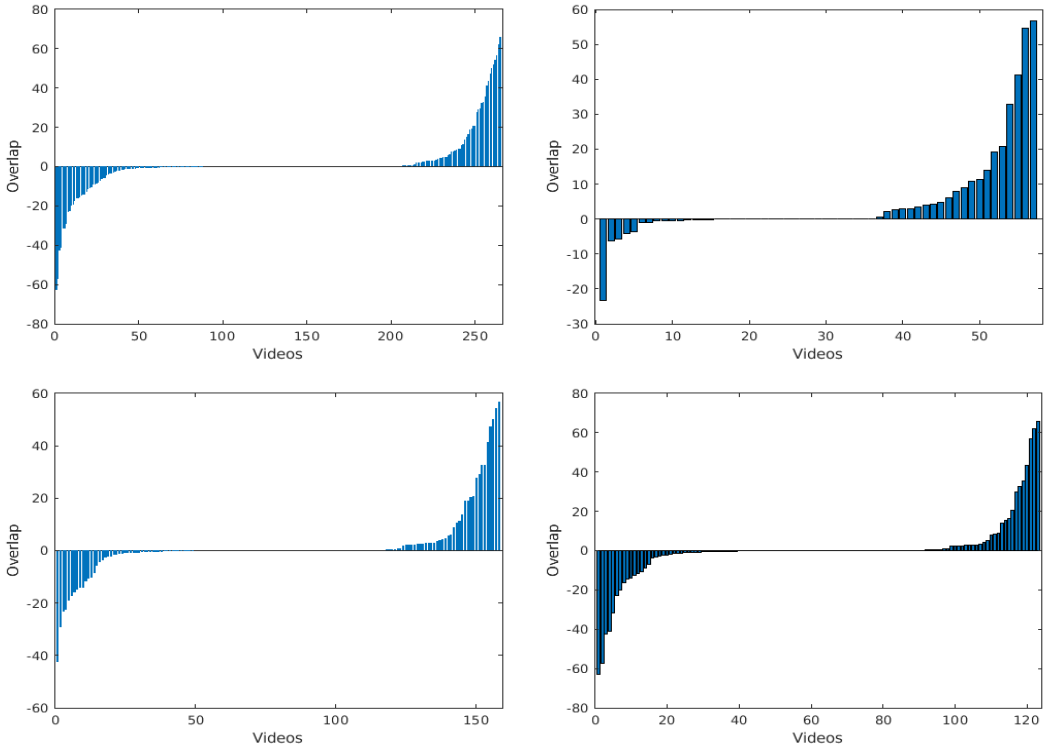


*Figura 22: Mejora en overlap (o IoU) para cada uno de los videos respecto del sistema original con los resultados ordenados de menor a mayor. Obtenido con el sistema de **backtracking acumulado** integrado sobre la red **SiamFc**. Resultados desglosados por categorías. Primera fila, primera imagen: resultados obtenidos para cada uno de los videos que conforman el dataset. Segunda imagen: resultados únicamente sobre videos con presencia de distractores. Segunda fila, primera imagen: resultados obtenidos sobre videos con presencia de oclusiones parciales. Segunda fila, segunda imagen: videos con presencia de oclusiones completas.*

CFNet	*Distancia*	*Overlap (IOU)*	*FPS*
Default	59.13	37.58	30.0
Backtracking Simple	**58.06**	**37.93**	8.0

Tabla 6: Comparativa de los resultados obtenidos sobre videos con presencia de oclusiones parciales empleando la red CFNet. A menor distancia y mayor IoU mejores resultados.

SIAM FC	*Distancia*	*Overlap (IOU)*	*FPS*
Default	65.19	**31.67**	33.0
Backtracking Simple	65.49	31.27	7.9
Backtracking Acumulado	**65.02**	31.54	2.4

Tabla 7: Comparativa de los resultados obtenidos sobre videos con presencia de oclusiones totales empleando la red SiamFC. A menor distancia y mayor IoU mejores resultados.

CFNet	*Distancia*	*Overlap (IOU)*	*FPS*
Default	**66.18**	**31.75**	30.0
Backtracking Simple	67.39	30.31	8.0

Tabla 8: Comparativa de los resultados obtenidos sobre videos con presencia de oclusiones totales empleando la red CFNet. A menor distancia y mayor IoU mejores resultados.

4.4.2 Videos con presencia de oclusiones totales

En la Tabla 7 y la Tabla 8 se puede observar cómo los resultados obtenidos en los videos en los que hay presencia de oclusiones totales los resultados empeoran respecto al sistema original tanto sobre la red SiamFC como sobre la red CFNet. Este fenómeno tiene sentido, puesto que, como se ha comentado anteriormente, si un objeto está completamente ocluido su camino desaparece y al realizar el *backtracking* el sistema no es capaz de converger en ninguna posición.

En la Figura 21, la Figura 22 y la Figura 23, en la cuarta imagen de cada una, que representan los videos con oclusiones totales, podemos ver como la cantidad de videos que empeoran es mucho mayor que en el resto de los casos y los que mejoran son muy pocos en comparación. Esto esta relacionado con la perdida de un candidato concreto que seguir (y por tanto de trayectoria) lo cual provoca que nuestro sistema falle.

4.4.3 Videos con presencia de distractores

En la Tabla 9 y la Tabla 10 se puede observar cómo los resultados obtenidos en los videos en los que hay presencia de distractores los resultados mejoran en gran medida, siendo la categoría que más lo hace. Esto es coherente con la aproximación seguida en este trabajo puesto que al añadir mediante estos sistemas un cierto grado de información temporal de las trayectorias seguidas es más sencillo distinguir entre dos objetos idénticos pero que han seguido trayectorias diferentes a lo largo de la secuencia (Figura 15). En la segunda imagen de la Figura 21, la Figura 22 y la Figura 23 podemos ver como prácticamente todos los videos con presencia de distractores mejoran, y en una proporción mucho mayor que en el caso de las oclusiones parciales. Además, a diferencia de en los casos anteriores, esta mejora es significativa sobre ambos *trackers* base tanto sobre SiamFc como sobre CFNet.

SIAM FC	*Distancia*	*Overlap (IOU)*	*FPS*
Default	75.04	21.22	33.0
Backtracking Simple	**69.05**	**25.90**	7.9
Backtracking Acumulado	69.63	25.25	2.4

Tabla 9: Comparativa de los resultados obtenidos sobre videos con presencia de distractores empleando la red SiamFC. A menor distancia y mayor IoU mejores resultados.

CFNet	*Distancia*	*Overlap (IOU)*	*FPS*
Default	71.94	23.28	30.0
Backtracking Simple	**68.52**	**25.62**	8.0

Tabla 10: Comparativa de los resultados obtenidos sobre videos con presencia de distractores empleando la red CFNet. A menor distancia y mayor IoU mejores resultados.

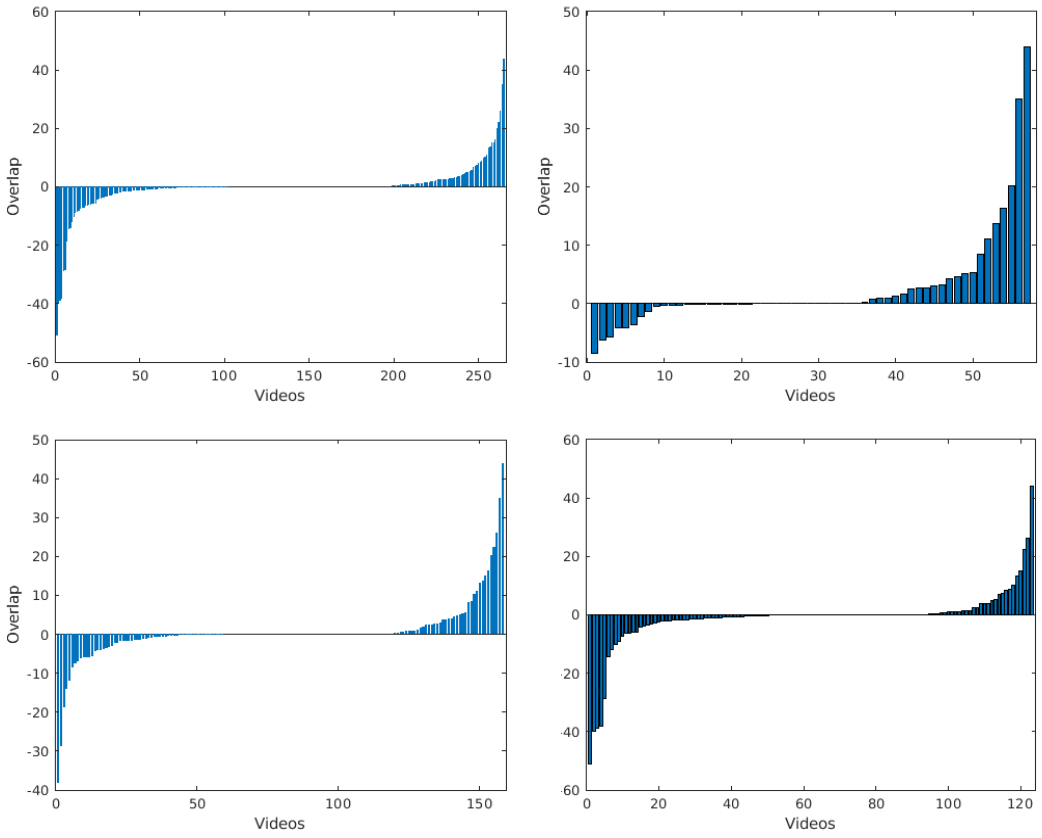


*Figura 23: Mejora en overlap (o IoU) para cada uno de los videos respecto del sistema original con los resultados ordenados de menor a mayor. Obtenido con el sistema de **backtracking simple** integrada sobre la red **CFNet**. Resultados desglosados por categorías. Primera fila, primera imagen: resultados obtenidos para cada uno de los videos que conforman el dataset. Segunda imagen: resultados únicamente sobre videos con presencia de distractores. Segunda fila, primera imagen: resultados obtenidos sobre videos con presencia de oclusiones parciales. Segunda fila, segunda imagen: videos con presencia de oclusiones completas.*

4.5 DISCUSIÓN DE LOS RESULTADOS

A partir de los resultados obtenidos en este apartado podemos observar cómo los sistemas creados en este trabajo mejoran el comportamiento del *tracker* ante la presencia de oclusiones parciales y distractores, pero da lugar a peores resultados ante la aparición de oclusiones completas. Esto, como ya se ha comentado, es un comportamiento esperado y una de las limitaciones inherentes de estos algoritmos puesto que ante la presencia de estas oclusiones completas y la desaparición del objeto no es posible seguir ninguna trayectoria hacia atrás y, en el caso del sistema de *backtracking* acumulado, puede provocar la pérdida del camino que se estaba siguiendo, provocando problemas a la hora de continuar con el seguimiento tras la reaparición del objeto.

Parece relevante también comentar los buenos obtenidos por estos sistemas ante la aparición de distractores en la escena, con grandes mejoras en ambos respecto del *tracker* original. Este resultado también parece coherente con el diseño empleado, puesto que distinguir entre dos objetos idénticos pero que han seguido trayectorias diferentes a lo largo del tiempo es relativamente arbitrario para este tipo de algoritmo.

Por último, comentar los resultados obtenidos por el sistema de backtracking simple sobre las redes SiamFc y CFNet. Hemos podido ver cómo, en casi todos los casos, las mejoras son más notables en el primer caso que en el segundo y si bien más análisis es necesario para poder indicar con precisión el motivo de esta diferencia hace dudar de hasta qué punto estos sistemas pueden llegar a generalizar bien sobre diferentes *tracker* sin necesidad de realizar una recalibración completa. Además, los resultados de velocidad (*fps*) obtenidos nos indican que, en un ámbito en el que el cálculo en tiempo real es especialmente importante, estos sistemas sufren una gran limitación debido a su poca eficiencia.

5 CONCLUSIONES Y TRABAJO FUTURO

En este trabajo se buscaba estudiar los diversos problemas que dificultan el proceso de seguimiento de objetos durante secuencias de video, así como diseñar una manera de hacer estos más robustos frente a ellos. Como se ha comentado anteriormente esto no es algo nuevo si no que diversos planteamientos diferentes se han seguido a lo largo del tiempo. Por ello en este trabajo nos hemos querido centrar en la creación e implementación de sistemas que cumplan este tipo de función en los *trackers* que conforman actualmente el estado del arte: los basados en redes neuronales siamesas.

Para ello en este trabajo se han estudiado e implementado dos sistemas diferentes qué, basándose en la técnica conocida como *backtracking*, buscan hacer más robustos frente a oclusiones y distractores los *trackers* basados en redes neuronales siamesas y mapas de correlación. Se ha comprobado experimentalmente que estas implementaciones mejoran los resultados obtenidos por estos en las secuencias que tienen presencia de oclusiones parciales y distractores pero que obtienen peores resultados en los casos en los que se dan oclusiones completas puesto que en estos casos no es posible seguir la trayectoria original del objeto lo que provoca que fallos en el algoritmo.

También se ha comprobado como estos sistemas, si bien ayudan a paliar estos problemas, tienen un gran impacto a nivel de eficiencia en el sistema bajando en ordenes de magnitud la velocidad de este lo cual es una gran limitación en su uso puesto que, como se ha comentado en otras ocasiones, en muchos casos es necesario el uso de estos en tiempo real (30 *fps*).

Es por todo esto que como trabajo futuro se plantean, al menos, los siguientes puntos:

- Integrar estos sistemas sobre las redes neuronales que conforman el estado del arte actual como pueden ser SiamRPN++ [3] o DASiam [2] y comprobar su comportamiento y resultados frente a estos problemas.
- Comprobar en mayor profundidad la versatilidad de estos módulos, puesto que los resultados obtenidos varían en gran medida dependiendo de la red empleada como base.
- Añadir formas de mitigar los problemas relacionados con las oclusiones completas que, ahora mismo, suponen una limitación a la generalización del sistema.

Buscar formas de mejorar la eficiencia del sistema de forma que sea acerca a la alcanzada por el *tracker* de referencia sobre el que se incluyen las mejoras.

6 REFERENCIAS

[1] J. Valmadre, L. Bertinetto, J. H. A. Vedaldi y P. H. S. Torr, «Ent-to-end representation learning for Correlation Filter based tracking,» *Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 2805-2813, 2017.

[2] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan y W. Hu, «Distractor-aware Siamese Networks for Visual Object Tracking,» *Proceedings of the European Conference on Computer Vision (ECCV),* pp. 101-117, 2018.

[3] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing y J. Yan, «Siamrpn++: Evolution of siamese visual tracking with very deep networks,» *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* pp. 4282-4291, 2019.

[4] B Y Lee et al, «Occlusion handling in videos object tracking: A survey,» *IOP conference series: earth and environmental science,* vol. 18, nº 1, p. 012020, 2014.

[5] Y. Lim, A. Gardi, N. Pongsakornsathien, R. Sabatini, N. Ezer y T. Kistan, «Experimental characterisation of eye-tracking sensors for adaptive human-machine systems,» *Measurement,* vol. 140, pp. 151-160, 2019.

[6] S. Ojha y S. Sakhare, «Image processing techniques for object tracking in video surveillance-A survey.,» *2015 International Conference on Pervasive Computing (ICPC),* pp. 1-6, 2015.

[7] V. Frati y D. Prattichizzo, «Using Kinect for hand tracking and rendering in wearable haptics.,» *IEEE World Haptics Conference,* pp. 317-321, 2011.

[8] S. Hua y D. C. Anastasiu, «Effective vehicle tracking algorithm for smart traffic networks,» *IEEE International Conference on Service-Oriented System Engineering (SOSE),* pp. 67-6709, 2019.

[9] P. Li y T. Qin, «Stereo vision-based semantic 3d object and ego-motion tracking for autonomous driving.,» *Proceedings of the European Conference on Computer Vision (ECCV),* pp. 646-661, 2018.

[10] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin y e. al, «The Visual Object Tracking VOT2018 Challenge Results,» 2018.

[11] M. Kristan, J. Matas, A. Leonardis y T. Vojir, «A Novel Performance Evaluation Methodology for Single-Target Trackers,» *IEEE transactions on pattern analysis and machine intelligence,* pp. 2137-2155, 2016.

[12] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin y e. al, «The Visual Object Tracking VOT2015 Challenge Results,» *Proceedings of the IEEE international conference on computer vision workshops,* pp. 1-23, 2015.

[13] X. Dong, J. Shen, D. Yu, W. Wang, J. Liu y H. Huang, «Occlusion-Aware Real-Time Object Tracking,» *IEEE Transactions on Multimedia,* pp. 763-771, 2017.

[14] X. Wang, Z. Hou, W. Yu, L. Pu, Z. Jin y X. Qin, «Robust occlusion-aware part-based visual tracking with object scale adaptation,» *Pattern Recognition 81,* pp. 456-470, 2018.

[15] J. Valmadre, L. Bertinetto, J. F. Henriques, R. Tao, A. Vedaldi, A. W. Smeulders, P. H. Torr y E. Gavves, «Long-term tracking in the wild: A benchmark,» *Proceedings of the European conference on computer vision (ECCV),* pp. 670-685, 2018.

[16] A. Yilmaz, O. Javed y M. Shah, «Object tracking: A survey.,» *Acm computing surveys (CSUR),* vol. 38, nº 4, pp. 13-es, 2006.

[17] P. Gabriel, J.-B. Hayet, J. Piater y J. Verly, «Object Tracking using Color Interest Points,» *IEEE Conference on Advanced Video and Signal Based Surveillance,* pp. 159-164, 2005.

[18] R. Damle y A. Gurjar, «Human Body Skeleton Detection and Tracking,» *International Journal of Technical Research and Applications,* pp. 2320-8163, 2015.

[19] Z. Pezzementi, S. Voros y G. D. Hager, «Articulated Object Tracking By Rendering Consistent Appearance Parts,» *IEEE International Conference on Robotics and Automation,* pp. 3940-3947, 2009.

[20] Z. Wang y X. Yang, «Moving target detection and tracking based on Pyramid Lucas-Kanade optical flow,» *IEEE 3rd International Conference on Image, Vision and Computing (ICIVC),* pp. 66-69, 2018.

[21] P. Bilinski, F. Bremond y M. B. Kaaniche, «Multiple Object Tracking with Occlusions using HOG Descriptors and Multi Resolution Images,» *3rd International Conference on Imaging for Crime Detection and Prevention,* pp. 1-6, 2009.

[22] D. S. Bolme, J. R. Beveridge, B. A. Draper y Y. M. Lui, «Visual Object Tracking using Adaptive Correlation Filters,» *IEEE computer society conference on computer vision and pattern recognition,* pp. 2544-2550, 2010.

[23] Y. Yuan, J. Chu, L. Leng y e. al, «A scale-adaptive object-tracking algorithm with occlusion detection,» *EURASIP Journal on Image and Video Processing,* pp. 1-15, 2020.

[24] G. Koch, «Siamese Neural Networks for One-Shot Image Recognition,» *ICML deep learning workshop,* vol. 2, 2015.

[25] X. Dong y J. Shen, «Triplet loss in siamese network for object tracking,» *Proceedings of the European conference on computer vision (ECCV),* pp. 459-474, 2018.

[26] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi y P. H. S. Torr, «Fully-Convolutional Siamese Networks for Object Tracking,» *European conference on computer vision,* pp. 850-865, 2016.

[27] B. Li, J. Yan, W. Wu, Z. Zhu y X. Hu, «High Performance Visual Tracking with Siamese Region Proposal Network,» *Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 8971-8980, 2018.

[28] S. Ren, K. He, R. Girshick y J. Sun, «Faster R-CNN: towards real-time object detection with region proposal networks,» 2016.

[29] R. Girshick, «Fast r-cnn,» *Proceedings of the IEEE international conference on computer vision,* pp. 1440-1448, 2015.

[30] L. A. Camuñas-Mesa, T. Serrano-Gotarredona, S.-H. Ieng, R. Benosman y B. Linares-Barranco, «Event-driven stereo visual tracking algorithm to solve object occlusion,» *IEEE transactions on neural networks and learning systems,* pp. 4223-4237, 2017.

[31] T. Liu, G. Wang y Q. Yang, «Real-time part-based visual tracking via adaptive correlation filters,» *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 4902-4912, 2015.

[32] Y. Xu, J. Wang, Y. Li, Z. Miao y Y. Zhang, «One-step backtracking for occlusion detection in real-time visual tracking,» *Electronics Letters,* vol. 53, nº 5, pp. 318-320, 2017.

[33] H. K. Galoogahi, A. Fagg, C. Huang, D. Ramanan y S. Lucey, «Need for Speed: A Benchmark for Higher Frame Rate Object Tracking,» *Proceedings of the IEEE International Conference on Computer Vision,* pp. 1125-1134, 2017.

[34] A. Li, M. Lin, Y. Wu, M. Yang y S. Yan, «NUS-PRO: A New Visual Tracking Challenge,» *IEEE transactions on pattern analysis and machine intelligence,* vol. 38, nº 2, pp. 335-349, 2016.

[35] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid y S. Savarese, «Generalized intersection over union: A metric and a loss for bounding box regression,» *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* pp. 658-666, 2019.

[36] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár y C. L. Zitnick, «Microsoft COCO: Common Objects in Context,» *European conference on computer vision,* pp. 740-755, 2014.

[37] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso y A. Torralba, «Scene Parsing through ADE20K Dataset,» *Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 633-641, 2017.

[38] R. Rout, «A survey on object detection and tracking algorithms.,» 2013.

[39] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin y e. al, «The Visual Object Tracking VOT2016 Challenge Results,» 2016.

[40] H.-Y. Cheng y J.-N. Hwang, «Integrated video object tracking with applications in trajectory-based event detection,» *Journal of Visual Communication and Image Representation,* vol. 22, nº 7, pp. 673-685, 2011.

ANEXO

A. GENERADOR DE OCLUSIONES

Si bien finalmente no ha sido utilizado en este trabajo, uno de los primeros planteamientos que se siguieron para intentar subsanar el problema comentado en el apartado 2.3.1.1 sobre la falta de variedad de oclusiones en los *dataset* fue la de generar de manera artificial imágenes ocluidas con distinto grado de oclusión a partir de objetos obtenidos de imágenes segmentadas semánticamente. De esta forma se puede generar un *dataset* teóricamente infinito de objetos ocluidos. Para ello se emplearon los siguientes *datasets* para extraer la imagen de diversos objetos que podrían ser empleados como oclusores:

- **COCO Dataset:** Contiene más de 330000 imágenes anotadas a nivel de objeto y de segmentación. Incluye 80 categorías de objetos. Si bien no contiene secuencias de video es útil a la hora de extraer objetos o imágenes individuales [36].

- **ADE20k Dataset:** *Dataset* para segmentación de escena con más de 20000 imágenes anotadas. Si bien está orientado a la segmentación semántica es útil también a la hora de extraer objetos de *frames* individuales debido a sus máscaras con precisión de píxel (no rectangulares). Estos objetos extraídos serán usados posteriormente para generar un aumento de datos en entrenamiento [37].

Este sistema funciona mediante la técnica conocida como *Alpha blending* que permite combinar una imagen con un fondo mediante el uso de la transparencia (empleando el canal *alpha* de la imagen, de ahí el nombre) entre una imagen y otra, de forma que la imagen se funde con el fondo dando una mayor sensación de integración en la imagen resultado.

$$\alpha_0 = \alpha_a + \alpha_b(1 - \alpha_a) \quad \textit{(Ecuación A.1)}$$

$$C_0 = \frac{C_a\alpha_a + C_b\alpha_b(1-\alpha_a)}{\alpha_0} \quad \textit{(Ecuación A.2)}$$

En las Ecuaciones A.1 y A.2 se puede ver la formula clásica del *alpha blending* donde para dos imágenes A y B con imagen resultado 0 los valores C_0, C_a y C_b representan los valores de los canales de color de los pixeles de cada una de las imágenes, donde la operación se aplica para cada canal de forma individual. Y α_0, α_a y α_b son los valores del canal *alpha* de cada una de las imágenes. Un par de ejemplos de imágenes obtenidas mediante este método se pueden ver en la Figura 24.

Figura 24: Ejemplo imágenes con oclusiones generadas mediante alpha blending.